Julius Schärf

BASIC für Anfänger

Fünfte, verbesserte Auflage

Programmieren leicht und schnell erlernbar

R. Oldenbourg Verlag Wien München

FORTBILDUNG DURCH SELBSTSTUDIUM

Herausgeber: Julius Schärf

Die Buchreihe

„FORTBILDUNG DURCH SELBSTSTUDIUM"

verhilft zu einer schnellen und sicheren Aneignung von Wissen, das für den beruflichen Aufstieg in der heutigen Leistungsgesellschaft erforderlich ist.

Anhand von vollständig durchgeführten Musterbeispielen dringt der Leser schrittweise in das Wissensgebiet ein. Viele Übungen mit angegebenen Lösungen geben ihm die Möglichkeit, die erworbenen Kenntnisse zu überprüfen und schon bald die ihm aufgetragenen Arbeiten zu bewältigen.

Prof. Dr. Julius Schärf wurde 1925 in Wien geboren. Er unterrichtet seit 1948 an Höheren technischen Lehranstalten. Er hat bereits 1957 die Fachgebiete Datenverarbeitung und technische Statistik, 1963 die Netzplantechnik in den Unterricht an der höheren Abteilung für Betriebstechnik am TGM eingeführt.

Dr. Schärf war einige Jahre in leitender Position in der Datenverarbeitungsindustrie tätig und ist Verfasser des vierbändigen Werks „Mathematik für Höhere technische Lehranstalten", einiger Lehrbücher und einiger Fachbücher.

JULIUS SCHÄRF

BASIC
FÜR ANFÄNGER

MIT EINEM GELEITWORT VON
KONRAD ZUSE

FÜNFTE, VERBESSERTE AUFLAGE

R. OLDENBOURG VERLAG WIEN MÜNCHEN 1977

Vom Bundesministerium für Unterricht und Kunst mit Erlaß vom 24. Oktober 1972 (Zl. 106.906-1/4b/72) für den IV. und V. Jahrgang der Höheren technischen und gewerblichen Lehranstalten mit Erlaß vom 22. Dezember 1976 (Zl. 24.933/1-14a/76) für die 5. bis 8. Klasse der Allgemeinbildenden höheren Schulen und mit Erlaß vom 4. Mai 1977 (Zl. 24.933/3-14a/76) für den V. Jahrgang der Höheren land- und forstwirtschaftlichen Lehranstalten zum Unterrichtsgebrauch zugelassen.

Schulbuch-Nr. **1272**

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Schärf, Julius

BASIC für Anfänger. – 5., verb. Aufl. — Wien, München: Oldenbourg, 1977.
(Fortbildung durch Selbststudium)
ISBN 3-486-39205-0 (München)
ISBN 3-7029-0115-9 (Wien)

1. Auflage 1970

© 1977 R. Oldenbourg KG Wien

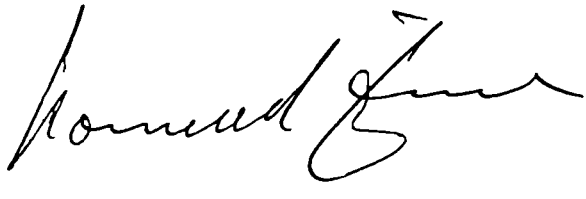
Das Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, der Funksendung, der Wiedergabe auf photomechanischem oder ähnlichem Wege sowie der Speicherung und Auswertung in Datenverarbeitungsanlagen, bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Werden mit schriftlicher Einwilligung des Verlags einzelne Vervielfältigungsstücke für gewerbliche Zwecke hergestellt, ist an den Verlag die gesetzlich zu zahlende Vergütung zu entrichten, über deren Höhe der Verlag Auskunft gibt.

Zeichnungen: Prof. Dipl.-Ing. F. Rohowsky
Druck: R. Spies & Co., Wien

ISBN 3-7029-0115-9 R. Oldenbourg Wien
ISBN 3-486-39205-0 R. Oldenbourg München

GELEITWORT

Die Bedeutung der elektronischen Datenverarbeitung braucht heute nicht mehr besonders betont zu werden. Leider hat sich aber noch immer nicht die Erkenntnis durchgesetzt, daß auch der Schulung auf diesem Gebiete erhebliche Bedeutung zukommt. Die Entwicklung begann damit, daß sich einige Außenseiter ganz diesen neuen Ideen widmeten und die ersten Grundlagen schufen. Dann wurde die Anwendung und Programmierung dieser Geräte eine Angelegenheit von Spezialisten. Heute kommen wir langsam dahin, daß fast ein jeder, der in der Wirtschaft, der Technik und der Wissenschaft tätig ist, mit der elektronischen Datenverarbeitung in Berührung kommt. Man kann daher nicht früh genug mit der Ausbildung auf diesem Gebiet beginnen. Es ist zu hoffen, daß das vorliegende Buch einen wichtigen Beitrag leisten wird, um möglichst weite Kreise mit den Fragen der Programmierung vertraut zu machen. Die Zukunft gehört denjenigen jungen Menschen, welche den technischen Fortschritt unserer Zeit am besten zu nutzen verstehen.



Professor Dr.-Ing. e. h. KONRAD ZUSE wurde am 22. Juni 1910 in Berlin geboren. 1941 vollendete er die erste programmgesteuerte Rechenanlage der Welt. Bereits 1945 hat er mit der Entwicklung der universellen algorithmischen Sprache PLANKALKÜL den Weg für symbolische Programmiersprachen gewiesen.

AUS DEM VORWORT ZUR ERSTEN AUFLAGE

In der Urzeit der Datenverarbeitung hat jede Anlage nur eine auf sie zugeschnittene Sprache verstanden. Der Programmierer mußte alle Operationen in kleinste Teile zerlegen und in verschlüsselter Weise dem Rechner mitteilen. Weil diese Art der Programmierung die Bewältigung größerer Aufgaben unmöglich machte, schuf man problemorientierte Programmiersprachen. Für wissenschaftliche und technische Probleme wurde 1954 in Amerika die Sprache FORTRAN und 1960 in Europa die Sprache ALGOL geschaffen. Für kaufmännische Probleme wurde 1960 die Sprache COBOL entwickelt, für kaufmännische und technische Probleme 1965 von der Firma IBM die Sprache PL/1.

In den USA erkannte man frühzeitig die Notwendigkeit, auch Schüler mit der Programmierung vertraut zu machen. Es wurde in Zusammenarbeit der National Science Foundation mit dem Dartmouth College die Sprache BASIC geschaffen. Diese Sprache nützt die großen technischen Möglichkeiten der modernen Datenverarbeitungsanlagen in erster Linie dahingehend aus, dem Anfänger das Programmieren zu erleichtern.

Im vorliegenden Buch wird nur jener Teil von BASIC behandelt, dessen Kenntnis für die Lösung von Problemen der technischen Praxis notwendig ist. Die meisten Beispiele wurden von etwa 18jährigen Schülern einer höheren technischen Lehranstalt erstellt. Auf elegante Druckanordnung und optimale Programmierung wurde im Hinblick auf die Tatsache, daß es sich hier um ein Buch für Anfänger handelt, aus pädagogischen Gründen absichtlich verzichtet.

Ich danke Herrn Prof. Dr. K. ZUSE für das Geleitwort. Ich danke allen Kollegen, die mir durch wertvolle Hinweise und Korrekturarbeiten bei der Gestaltung dieses Buches geholfen haben, besonders den Herren Dr. G. DEMMELBAUER und Dr. R. FOHRINGER.

VORWORT ZUR VIERTEN AUFLAGE

Vor knapp drei Jahren erschien die erste Auflage dieses Buchs unter dem Titel „Programmieren leicht und schnell erlernbar; Einführung in BASIC". Diese Sprache war damals im deutschen Sprachraum wenig bekannt. Die von mir gewählte Methode, den Leser unter Verzicht auf unnötigen Ballast schnellstens zum Programmieren zu bringen, war Neuland.

Die sehr gute Aufnahme des Buchs bewog den Verlag, es vor etwa zwei Jahren zum Ausgangspunkt einer für das Selbststudium bestimmten Serie zu machen, in der bereits weitere Bände erschienen sind.

Der grundsätzliche Aufbau wurde beibehalten, Vereinfachungen der Sprache wurden berücksichtigt. Zusätzlich wurden am Beginn jedes Abschnitts die Lernziele angegeben. Bei den angeführten Lernzielen handelt es sich um Mindestanforderungen, die sich auf elementare Fähigkeiten beziehen. Es genügt jedoch nicht, daß der Lernende über diese Fähigkeiten nur getrennt voneinander verfügen kann. Wesentlich ist vielmehr, daß die Fähigkeiten auch bei der Lösung komplexer Aufgaben in koordinierter Weise zum Ausdruck kommen. Die Wahl der Aufgaben in diesem Buch wurde nach diesem Gesichtspunkt getroffen.

Um den Leser schon beim Studium an das für ihn neue Schriftbild des Terminals zu gewöhnen, wurden Programme und Ausgabeprotokolle fotografisch wiedergegeben.

Die in diesem Buch verwendete Teilmenge von BASIC entspricht dem General-Electric Betriebssystem Mark I,7 und kann nicht nur bei Time-Sharing-Anlagen verwendet werden, sogar Tischrechner verstehen diese Sprache.

Auf das Betriebssystem Mark III wurde nicht eingegangen, die Besprechung dieses Systems erfolgt im Folgeband „BASIC-Praxis" von H. O. Ramp.

Die Lernziele wurden der vom Bundesministerium für Unterricht und Kunst geschaffenen LEHRZIELBANK FÜR DAS BERUFSBILDENDE SCHULWESEN, Elektronische Datenverarbeitung für Höhere technische Lehranstalten, entnommen.

Ich danke für Verbesserungsvorschläge, insbesondere den Herren Dr. Helmut Aigner, Dr. Franz Blaha, Hubert Horstmann, Dipl.-Ing. Herbert Ramp, Mag. Robert Strecha und Dr. Tobias Schläpfer, und bitte alle Leser um Kritik und Anregungen.

Zu Dank verpflichtet bin ich ferner der Firma Honeywell-Bull, Wien, für die gewährte Hilfe und dem Verlag R. Oldenbourg für das Eingehen auf meine Wünsche.

Wien, Juni 1975

In der vorliegenden **5. Auflage** wurden Berichtigungen und Präzisierungen vorgenommen.

Fast alle Vorschläge zur 4. Auflage konnten so realisiert werden.

Wien, Juni 1977 *Julius Schärf*

HINWEISE FÜR DEN LESER

Das vorliegende Buch hat zwei Ziele:

1. Elementare Einführung von Anfängern in das Programmieren.
2. Vertiefung der numerischen Mathematik.

Es werden keine speziellen Mathematikkenntnisse vorausgesetzt. Im allgemeinen genügt der Stoff von 10 Schulstufen.

Jede Programmierregel wird an mehreren Beispielen erläutert. Es können daher Leser, die auf eine Vertiefung ihrer Mathematikkenntnisse keinen Wert legen, Beispiele und Übungen, die einen größeren mathematischen Aufwand erfordern, überspringen. Das Ziel, in kurzer Zeit programmieren zu lernen, wird dadurch nicht beeinträchtigt.

Aufbau des Buchs

Im Abschnitt 1.3. wird der Leser mit der Programmierung vertraut gemacht. Es ist unbedingt notwendig, die Beispiele gewissenhaft durchzuarbeiten, die Übungen auszuführen und zu überprüfen, ob die Ergebnisse richtig sind.

Wenn weniger als 11 der 14 Übungen richtig beantwortet wurden, ist das neuerliche Durcharbeiten dieses Abschnitts notwendig.

In den Abschnitten 2.—8. werden die bereits erworbenen Kenntnisse vertieft und ausgebaut.

Viele vollständig ausgeführte Beispiele und Übungen mit vollständig angegebener Lösung sorgen für eine gründliche Erfassung des dargebotenen Stoffs.

Ausdrücklich sei darauf hingewiesen, daß die meisten in der technischen Praxis auftretenden Aufgaben bereits mit den Kenntnissen der Abschnitte 1.—8. gelöst werden können.

In den Abschnitten 9.—12. werden Anweisungen besprochen, die das Programmieren erleichtern, aber nicht unbedingt notwendig sind.

Im Abschnitt 13 wird das Wissen an Hand einiger ausführlich kommentierter Beispiele vertieft.

Anschließend werden die wichtigsten Systembefehle des Betriebssystems Mark I/7 und die EDIT-Befehle besprochen.

Den Abschluß des Buchs bilden die vollständig angegebenen Lösungen zu den Übungen.

INHALTSVERZEICHNIS

1. Einführung 11

1.1. Was ist eine Datenverarbeitungsanlage? 11
1.2. Was ist Time-Sharing? 14
1.3. Einführende Beispiele 19

2. Grundelemente von BASIC 31

2.1. Zeichen 32
2.2. Numerische und alphamerische Information 32
2.3. Numerische Konstanten 32
2.3.1. Ganzzahlen 32
2.3.2. Fixkommazahlen 33
2.3.3. Gleitkommazahlen 33
2.4. Numerische Variablen 34
2.5. Zeichen für Rechenoperationen 35
2.6. Reihenfolge von Rechenoperationen 36
2.7. Alphamerische Konstanten 37
2.8. Alphamerische Variablen 37
2.9. GOTO-Anweisung 38
2.10. STOP-Anweisung 38
2.11. END-Anweisung 38
2.12. REM-Anweisung 38

3. Terme 39

4. LET-Anweisung (ERGIBT-Anweisung) 42

5. Funktionen 48

5.1. Standardfunktionen 48
5.1.1. Quadratwurzelfunktion SQR 49
5.1.2. Absolutwertfunktion ABS 49
5.1.3. Vorzeichenfunktion SGN 49
5.1.4. Logarithmusfunktion LOG 49
5.1.5. Exponentialfunktion EXP 49
5.1.6. Kreisfunktionen SIN, COS, TAN 49
5.1.7. Arkustangensfunktion ATN 50
5.1.8. Integerfunktion INT 53
5.1.9. Randomfunktion RND 54
5.2. Vereinbarte Funktionen 57

6. Eingabe und Ausgabe 60
6.1. READ-Anweisung und DATA-Anweisung 60
6.2. INPUT-Anweisung 61
6.3. PRINT-Anweisung 61
6.4. Formatisierte Ausgabe 63

7. Programmablaufplan 65

8. Verzweigungen und Schleifen 68
8.1. IF-Anweisung 68
8.2. FOR-Anweisung und NEXT-Anweisung 77

9. Variablenfelder 85
9.1. Indizierte Variablen 85
9.2. Dimensionierung von Feldern 86

10. Zeichnen von Funktionskurven 91

11. Matrizenoperationen 95
11.1. Begriff der Matrix 95
11.2. Lesen und Drucken von Matrizen 96
11.3. Gleichheit, Addition und Subtraktion von Matrizen 98
11.4. Multiplikation einer Matrix mit einer Zahl 99
11.5. Transponierte Matrix 99
11.6. Einheitsmatrix 100
11.7. Nullmatrix, Konstante, Matrix 102
11.8. Multiplikation von Matrizen 103
11.9. Inverse Matrix 104

12. Unterprogramme 110

13. Weitere Programmierungsbeispiele 118
Quadratische Gleichung 118
Kubische Gleichung 120
Reelle Nullstellen einer Funktion 124
Numerische Integration (Trapez-Regel) 126
Sortierverfahren 128
Berechnung von Querschnittswerten 132
Lineares Gleichungssystem (Verfahren von Gauß-Jordan) 135

14. Systembefehle 139

15. EDIT-Befehle 140

Lösungen zu den Übungen 142

Wichtige Konstanten 152

Register 153

1. EINFÜHRUNG

Lernziele

Den allgemeinen Aufbau einer EDVA angeben können.

Den allgemeinen Aufbau der Zentraleinheit einer EDVA angeben können.

Die Begriffe *on-line* und *off-line* erklären können.

Die Begriffe *Time-Sharing* und *Terminal* erklären können.

Den allgemeinen Aufbau einer Time-Sharing-Anlage angeben können.

Die Begriffe *Programm*, *Quellenprogramm*, *Übersetzerprogramm*, *Objektprogramm* und *Programmiersprache* erklären können.

Den Aufbau eines Lochstreifens erklären können.

Den Aufbau einer Programmzeile angeben können.

1.1. WAS IST EINE DATENVERARBEITUNGSANLAGE?

Wie schon der Name sagt, hat eine *Datenverarbeitungsanlage*[1], auch kurz *Computer* oder *Rechner* genannt, die Aufgabe, Daten zu verarbeiten.

Daten können Zahlen oder Wörter sein. Daten verarbeiten kann demnach Rechnen sein, aber auch Namen vergleichen und nach dem Alphabet sortieren, Listen drucken usw.

Damit die DVA Daten verarbeiten kann, muß sie diese lesen und die Ergebnisse drucken können[2].

Eine Datenverarbeitungsanlage besteht grundsätzlich aus 3 Teilen, nämlich aus

1. *Eingabegeräten*
2. *Zentraleinheit*
3. *Ausgabegeräten*

---

[1] Abkürzungen: DVA bzw. EDVA (elektronische Datenverarbeitungsanlage).

[2] In der Datenverarbeitung bezeichnet man jedes maschinelle Beschriften von Papier als „drucken".

Die Eingabe kann im einfachsten Fall, wie bei einer elektrischen Tischrechenmaschine, über eine Tastatur erfolgen. Wegen der hohen Rechengeschwindigkeiten der modernen Computer ist diese Eingabeart im allgemeinen unwirtschaftlich. Man benutzt daher Lochkarten, Lochstreifen oder Magnetbänder als Datenträger für die Eingabe. Die Daten können mit Geräten, die unabhängig vom Rechner sind, erfaßt werden. Man bezeichnet Geräte dieser Art als *off-line-Peripherie*, im Gegensatz zur *on-line-Peripherie*[1].

Die Ausgabe von Daten kann im einfachsten Fall über eine elektrische Schreibmaschine erfolgen[2]. In der kommerziellen Datenverarbeitung fallen viele Daten an. Es müssen daher dort Schnelldrucker als Ausgabegeräte eingesetzt werden[3].

Die Zentraleinheit besteht aus 3 Teilen:

1. *Hauptspeicher*[4]
2. *Steuerwerk*[5]
3. *Rechenwerk*[6]

Der *Hauptspeicher* ist das Gedächtnis der DVA. In ihm werden alle Befehle und Daten gespeichert, die der Rechner zur Durchführung einer Aufgabe benötigt. Allen Speicherstellen sind umkehrbar eindeutig Adressen zugeordnet.

Ausdrücklich sei festgehalten, daß der Hauptspeicher nicht nur Daten enthält. Vor Beginn einer Rechnung werden außer den benötigten Daten die *Rechenanweisungen* in den Hauptspeicher gebracht. Die Folge aller Anweisungen für eine bestimmte Aufgabe nennt man *Programm*.

Das *Steuerwerk* hat die Aufgabe, den Ablauf der einzelnen Anweisungen zu steuern.

Es muß also die Anweisungen

a) aus dem Speicher abrufen,

b) entschlüsseln,

c) in Steuersignale umsetzen,

somit die Befehlsausführung einleiten und überwachen.

Das *Rechenwerk* führt die arithmetischen und die logischen Operationen durch.

Auf der nächsten Seite ist eine große EDVA abgebildet.

---

[1] Das sind Geräte, die ständig mit dem Rechner verbunden sind. Beachte: Manche periphere Geräte können sowohl im ON-LINE-Betrieb als auch im OFF-LINE-Betrieb verwendet werden.

[2] bzw. über einen Fernschreiber mit etwa 10 Zeichen/Sekunde.

[3] Schnelldrucker bewältigen die Ausgabe von 20 und mehr Zeilen (mit je 120 Zeichen) pro Sekunde.

[4] engl.: central storage bzw. central memory

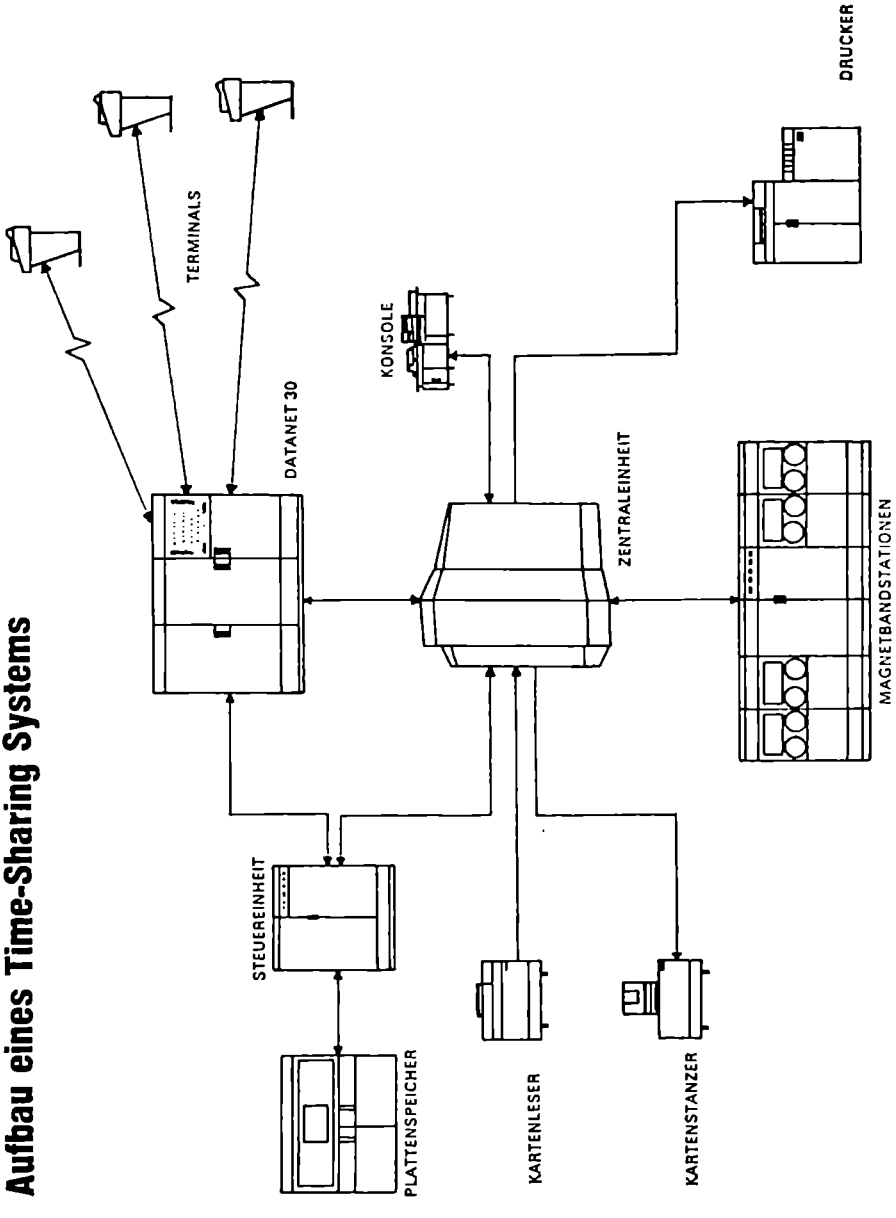
[5] engl.: control unit

[6] engl.: arithmetic unit

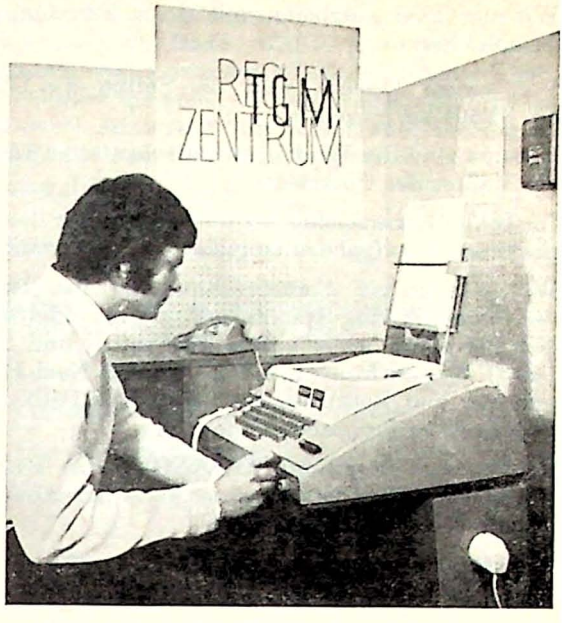
Zentraleinheit
Datenfernübertragung
Konsolschreibmaschine
Magnetbandeinheiten
Hauptterminal
Schnelldrucker
Kartenstanzer
Plattenspeicher

1.2. WAS IST TIME-SHARING?

Unter *Time-Sharing* („Zeitzuteilung") versteht man die gleichzeitige Verwendung eines Großcomputers durch mehrere Benützer, denen, z. B. über Fernsprechleitungen, die Rechenanlage abwechselnd für kurze Zeit (z. B. 3 Sekunden lang) zur Verfügung steht. Infolge der Schnelligkeit des Rechners im Vergleich zu den langsamen Ein- und Ausgabegeräten hat jeder Benützer den Eindruck, daß der Rechner ausschließlich ihm zur Verfügung steht.



Aufbau eines Time-Sharing Systems



Ein Schüler der Höheren Abteilung für Betriebstechnik am TGM bei der Arbeit am Fernschreib-Terminal.

Terminals (Datenplätze) sind Geräte, die zur Eingabe/Ausgabe von Daten dienen. Es werden für diesen Zweck sowohl geräuschlos arbeitende Bildschirmgeräte (Nachteil: keine schriftlichen Unterlagen!) als auch Fernschreib-Terminals verwendet.

Mit dem oben abgebildeten Teletype 33, dem am meisten verbreiteten Gerät, kann man auch Lochstreifen herstellen.

An große Time-Sharing-Anlagen können mehr als 100000 Terminals angeschlossen werden. Etwa 2000 dieser Teilnehmer können gleichzeitig mit dem Rechner arbeiten, ohne einander merklich zu behindern.

Zum Aufbau des Time-Sharing-Systems:

Dem *Hauptrechner* ist ein Datenübertragungsnetz vorgeschaltet.

Für jeden Teilnehmer sind im Datenübertragungsnetz ein *Eingabepuffer* und ein *Ausgabepuffer* vorgesehen.

Jedem Teilnehmer stehen folgende Plattenspeicherbereiche zur Verfügung:

Arbeitsspeicher: Speicher, in dem das Programm entweder über Terminaleingabe oder aus dem Bibliotheksspeicher zur Ausführung kommt.

Bibliotheksspeicher: In ihm sind die Programme und Daten der Teilnehmer und die *Systembibliothek* — für alle Teilnehmer zugängliche Programme — gespeichert.

Wie einfach das Arbeiten mit einem Terminal ist, geht aus dem folgenden Beispiel hervor.

Von einem Dreieck sind die Seiten $a = 17{,}782$ cm, $b = 4{,}635$ cm und $c = 20{,}501$ cm gegeben.

Gesucht sind die Winkel, der Flächeninhalt, der Radius des Inkreises und der Radius des Umkreises.

Im Bibliotheksspeicher befindet sich unter der Bezeichnung TRIANG*** das unserer Aufgabe entsprechende Programm.

Wir wählen die Fernsprechnummer des Rechenzentrums. Wenn sich das Time-Sharing-System mit hohem Pfeifton meldet, verbindet man den Fernschreiber mit dem TS-System und legt den Telephonhörer auf. Das TS-System fragt: USER NUMBER. Nach Eintasten unserer Teilnehmernummer und Betätigen der Taste RETURN fragt der Rechner nach der gewünschten Programmiersprache.

Bei unserem Programm TRIANG*** ist BASIC als Sprache angegeben. Wir teilen daher dem Rechner BAS (als Abkürzung für BASIC) mit und betätigen die Taste RETURN.

Das System fragt: NEW OR OLD.

Weil wir mit einem bereits vorhandenen, also alten Programm arbeiten wollen, tasten wir OLD, eine Leerstelle und den Programmnamen TRIANG*** ein. Dann betätigen wir die Taste RETURN.

Der Systemrechner veranlaßt, daß nun das Programm TRIANG*** aus dem Bibliotheksspeicher in den Arbeitsspeicher gebracht wird.

Die Meldung READY zeigt an, daß der Rechner zur Aufnahme von Systembefehlen oder Programmzeilen bereit ist.

```
HONEYWELL BULL TIME-SHARING

ON AT   8:51    WIEN  28/05/73    TTY  3
USER NUMBER--W17000
SYSTEM--BAS
NEW OR OLD--OLD TRIANG***


READY.

RUN
```

[1] *Beachte:* Die vom Benutzer des Terminals, dem *Operator*, eingegebenen Texte wurden zum besseren Verständnis nachträglich unterstrichen.

Der Systembefehl RUN bewirkt, daß das Programm vom Systemrechner an den Hauptrechner weitergegeben wird. Das in BASIC geschriebene Programm (*Quellenprogramm* genannt) wird zusammen mit einem BASIC-Übersetzungsprogramm (BASIC-*Compiler* genannt) in den Hauptspeicher des Hauptrechners gebracht und durch letzteres in die eigentliche Maschinensprache umgewandelt. Dieses Programm heißt *Objektprogramm.*
Sodann wird das Programm ausgeführt.

```
TRIANG      8:51    WIEN  28/05/73

DIESES HONEYWELL BULL TIME-SHARING DEMONSTRATIONSPROGRAMM
BERECHNET EIN DREIECK,WENN GEGEBEN SIND EINE SEITE
UND ZWEI ANDERE TEILE

WELCHE WERTE MOECHTEN SIE EINGEBEN?(1=SSS,2=SAS,3=SSA,4=ASA,5=AAS)? 1

BITTE GEBEN SIE DIE WERTE FUER SEITE1,SEITE2UND SEITE3 EIN
?  17.782,  4.635,  20.501

                    1            2            3
                 -------      -------      -------

SEITE             17.782        4.635       20.501

WINKEL BOG.       .848958       .196917     2.09572

WINKEL GRD.       48            11          120
      (MIN)       38            16          4
      (SEC)       30.19         57.07       32.74

HOEHE DER SEITE 4.01096        15.3879      3.479


RADIUS DES UMKREISES= 11.8453
RADIUS DES INKREISES= 1.66184
DIE FLAECHE BETRAEGT= 35.6614


EINE NEUE BERECHNUNG? (1=JA)? 0



USED     11.33 UNITS.                          ...1
BYE                                            ...2

*** OFF AT  8:55    WIEN  28/05/73

      4 ELAPSED TERMINAL MINUTES               ...3
    11.33 TOTAL CRU'S USED                     ...4
```

[1] Rechen- und Übersetzungszeit.
[2] Nach Beendigung der Rechnung wird durch den Systembefehl BYE die Datenstation vom Rechner getrennt.
[3] Gesamte Anschaltzeit.
[4] Gesamte Hauptrechnereinheiten.

Wir haben nun gesehen, wie leicht man mit Hilfe eines Time-Sharing-Systems verschiedene Probleme lösen kann. Um den Rechner voll ausnützen zu können, müssen wir ihm unsere Wünsche in einer Sprache bekanntgeben, die er versteht. Unser DV-System wurde von seinen geistigen Vätern mehrsprachig geschaffen.

Es versteht die problemorientierten Programmiersprachen

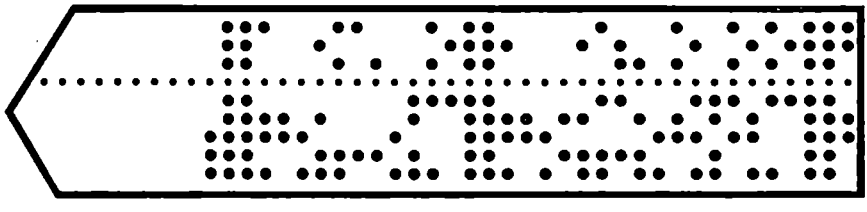
ALGOL	(Algorithmic Language)
FORTRAN	(Formula Translation)
BASIC	(Beginner's All-purpose Symbolic Instructions Code)

Wir haben uns, wie bereits im Vorwort angeführt wurde, für die einfachste und eleganteste dieser Sprachen, für BASIC, entschieden.

Bevor wir lernen, wie man ein Programm in BASIC schreibt, wollen wir noch kurz aufzeigen, wie man mit selbsterstellten Programmen arbeitet.

Man schreibt zunächst das Programm und die Ausgangsdaten am Blattschreiber des Terminals[1].

Diese Daten werden bei eingeschaltetem Lochstreifenstanzer in Form verschlüsselter Lochkombinationen in einen 1 Zoll breiten Papierstreifen, den *Lochstreifen*, gestanzt[2].



Nach genauer Prüfung des Programms auf mögliche Schreibfehler und Korrektur derselben legen wir den Lochstreifen in den Abtaster. Wir stellen die Verbindung Terminal–Systemrechner her und geben durch Eintasten unsere Teilnehmernummer und die von uns gewünschte Programmiersprache BASIC bekannt. Die Frage NEW OR OLD beantworten wir mit NEW, gefolgt von einer Leerstelle und dem von uns zu wählenden Programmnamen. Dieser muß mit einem Buchstaben beginnen und aus 1 bis 8 BASIC-Zeichen (ausgenommen Leerstellen) bestehen. Nach der Rechnermeldung READY geben wir mit dem Systembefehl TAPE (wir schreiben TAP) bekannt, daß wir nicht über die Tastatur, sondern über Lochstreifen einzugeben wünschen. Nach der neuerlichen Meldung READY schalten wir den Lochstreifenleser ein.

[1] *Beachte:* Eine DVA kann mit dem Programm allein im allgemeinen nichts anfangen. Es nützt z. B. nichts, der Rechenanlage zu befehlen: „Löse ein lineares Gleichungssystem in 82 Variablen", wenn man dem Rechner nicht die Koeffizienten dieses Systems bekanntgibt.

[2] *Beachte:* Ungestanzte Abschnitte des Lochstreifens bewirken sofortigen Abbruch des unmittelbar vorher gegebenen Systembefehls.

Unser Programm wird eingelesen und gleichzeitig geschrieben. Nach dem Einlesen schaltet der Lochstreifenleser selbsttätig ab. Wir setzen dann mit dem Systembefehl RUN fort.

1.3. EINFÜHRENDE BEISPIELE

1. Das Programm

```
10 PRINT 7+6
20 PRINT 9-15
30 PRINT 7*3
40 PRINT 64/16
```

Es liegt hier ein BASIC-Programm für die Berechnung von $7+6$, $9-15$, $7 \cdot 3$ und $64:16$ vor!

und der Systembefehl

RUN

bewirken die Ausgabe:

```
 13
-6
 21
 4
```

Beachte:

1. Jede Programmzeile beginnt mit einer *Zeilennummer.*

   Jede Zeilennummer besteht aus 1 bis 5 Ziffern, muß am linken Zeilenrand beginnen und darf keine Leerstelle zwischen den Ziffern aufweisen.

   Haben zwei Anweisungen eines Programms die gleiche Zeilennummer, so wird die vom Computer zuerst gelesene Anweisung durch die zweite ersetzt[1].

2. Nach dieser Zeilennummer steht jeweils eine *Anweisung.* Die Anweisungsworte sind der englischen Sprache entnommen.

   PRINT hat in BASIC die Bedeutung „(RECHNE und) *DRUCKE*".

   Die Anweisungen eines Programms werden in der Reihenfolge zunehmender Zeilennummern ausgeführt.

3. Das Betätigen der RETURN-Taste nach jeder Programmzeile bewirkt, daß der Rechner den Schreibkopf des Fernschreibers in seine Ausgangslage (Position 0) bringt und dann das Papier am Fernschreiber (Ausgabeprotokoll) weiterschiebt. Man spricht von einem *Zeilenvorschub.*

4. In BASIC wird * als Multiplikationszeichen und / als Divisionszeichen verwendet.

5. Bei dem hier verwendeten Terminal wird die Ziffer 0 durchgestrichen, also Ø geschrieben, um nicht mit dem Buchstaben O verwechselt zu werden.

6. Im Programm werden keine Kleinbuchstaben verwendet.

[1] Eine Anweisung kann somit gelöscht werden, indem man später eine Leeranweisung mit gleicher Zeilennummer schreibt!

7. Der Systembefehl RUN bewirkt, daß das Programm nach steigenden Zeilennummern geordnet, in die Maschinensprache übersetzt und dann ausgeführt wird.

8. Es ist ratsam, die Zeilennummern mit Intervallen von 10 zu wählen, um eventuell später weitere Anweisungen einfügen zu können.

Auf zwei weitere Systembefehle soll jetzt schon hingewiesen werden[1]:

1. Das Schreiben von LISTNH und anschließendes Betätigen der RETURN-Taste bewirkt das (nochmalige) Schreiben aller Programmzeilen.

2. Das Schreiben von EDI DEL 10—40 und anschließendes Betätigen der RETURN-Taste bewirkt das Löschen aller Programmzeilen von 10 bis einschließlich 40. . . .[2]

2.
```
10 PRINT 7+6, 9-15, 7*3, 64/16
```

oder

```
10 PRINT 7+6,
20 PRINT 9-15,
30 PRINT 7*3,
40 PRINT 64/16
```

bewirkt folgende Ausgabe:

Beachte: Die Beistriche nach den Zahlentermen[3] im Programm bewirken, daß die errechneten Werte in eine Zeile des Ausgabeprotokolls geschrieben werden[4].

```
 13             -6             21             4
```

3.
```
10 PRINT 8
20 PRINT
30 PRINT 5*3, 9*7,
40 PRINT 2+3-5*4
```

bewirkt folgende Ausgabe:

```
 8

 15             63             -15
```

Beachte: Die Zeile 10 schließt ohne Komma, daher wird nach dem Ausdrucken der Zahl 8 ein Zeilenvorschub durchgeführt.

Die Anweisung 20 PRINT bewirkt eine Leerzeile.

Zeile 30 schließt mit einem Komma, es wird an dieser Stelle kein Schreibkopfrücklauf und kein Zeilenvorschub durchgeführt.

---

[1] Systembefehle werden auf Seite 107 f. besprochen.

[2] EDI DEL ist eine Abkürzung für EDIT DELETE (edit △ aufbereiten; delete △ streichen).

[3] In der Mathematik bezeichnet man Kombinationen aus Zahlen und Rechenzeichen als *Terme*, in der Datenverarbeitung oft als *„numerische Ausdrücke"*.

[4] Durch Druckanweisungen dieser Art können 5 Zahlen in eine Zeile geschrieben werden; der Rechner teilt also eine Zeile in 5 Spalten.

4. Das Programm

```
10 READ X
20 PRINT X*X
30 PRINT X↑2
40 PRINT X↑3
50 PRINT X↑(1/2)
60 DATA .5
```

bewirkt die Ausgabe

```
 .25
 .25
 .125
 .707107
```

Bemerkung: 10 READ X befiehlt dem Rechner das Lesen einer Zahl, und zwar das Lesen der ersten Konstanten der DATA-Zeile[1].

Nach der Ausführung der in der Zeile 10 stehenden Anweisung führt der Rechner die nächste Anweisung aus.

Der Speicherplatz X ist bereits mit der Zahl 0,5 belegt. Der Computer rechnet daher $0{,}5 \cdot 0{,}5$ und druckt den Termwert 0,25 aus, veranlaßt den Rücklauf des Schreibkopfs in die Position Ø und führt einen Zeilenvorschub durch.

Der Rechner führt nun die nächste Anweisung, die in Zeile 30 stehende Anweisung PRINT X ↑ 2, aus und führt Schreibkopfrücklauf und einen Zeilenvorschub durch.

Beachte: In BASIC wird x^2 durch X ↑ 2 dargestellt.

Für x^n (n sei eine natürliche Zahl) schreibt man X ↑ N

Für $\sqrt[n]{x}$, wobei ein nichtnegativer Radikand x vorausgesetzt ist, schreibt man X ↑ (1/N)

Somit schreibt man X ↑ (1/2) für $\sqrt{x}$ und X ↑ (1/3) für $\sqrt[3]{x}$.

Für $\sqrt{x}$ kann man statt X ↑ (1/2) auch SQR(X) schreiben. SQR ist die Abkürzung für das englische Wort „square root".

Anschließend wird x^3 berechnet und gedruckt, wobei der Speicherplatz X mit Ø.5 belegt ist.

---

[1] READ X bedeutet hier demnach: *LIES* die erste Zahl der DATA-Zeile (in unserem Fall also die Zahl 0,5) und speichere sie unter dem Namen X.
Einfachheitshalber bezeichnen wir den für diese Speicherung vorgesehenen Speicherplatz kurz als Speicherplatz X.
data (engl.) ≙ Daten

Nach einem Zeilenvorschub wird als letzte Anweisung die in Zeile 50 stehende Anweisung PRINT X↑(1/2), also das Berechnen und Ausdrucken von $\sqrt{0.5}$, durchgeführt.

Die Zeile 60, die eine nicht ausführbare Anweisung enthält, wird bei der Programmausführung übersprungen. Da keine weitere Anweisung folgt, wird die Programmausführung beendet.

Beachte:

Dem bei uns üblichen Dezimalkomma entspricht in BASIC ein Punkt.

Bei unserem Terminal wird die Ziffer Null, zur deutlichen Unterscheidung gegenüber dem Buchstaben O, durchgestrichen.

Die Null vor dem Komma wird vom Rechner nicht angeschrieben.

5.
```
10 READ A,B
20 PRINT A,B,A+B,A*B,A/B
30 DATA 4, 5
```

Es liegt hier ein vollständiges Programm für die Berechnung von $4+5$ und $4 \cdot 5$ und $4:5$ vor.

Wie in der Mathematik werden auch in BASIC Rechnungen mit gleicher Struktur durch Terme mit Variablen dargestellt.

In der Zeile 10 steht die Leseanweisung READ. Durch sie werden die Variablen A und B mit den in der Zeile 30, in der DATA-Zeile, stehenden Zahlen 4 und 5 belegt. Der ersten Variablen A wird die erste Zahl, also 4, zugeordnet, der zweiten Variablen B die zweite Zahl, also 5.

In der Zeile 20 steht die Druckanweisung

```
20 PRINT  A, B, A+B, A*B, A/B
```

Durch sie wird der Rechner veranlaßt, die Zahlen

4 5 9 20 0,8 auszudrucken[1].

Das Ausgabeprotokoll hat folgende Form:

```
4             5             9             20            .8
```

[1] Wenn die in einer PRINT-Anweisung stehenden Variablen durch Kommata getrennt sind, so wird die Druckzeile in 5 Spalten zu je 15 Zeichen geteilt.

6. Im fünften Beispiel hat der Rechner die Zeile

```
 4            5            9            20           .8
```

ausgegeben. Wir wollen nun dieses Beispiel so erweitern, daß die Ausgabe folgende Form hat:

```
BEISPIEL 6
 A            B            A+B          A*B          A/B
 4            5            9            20           .8
```

Lösung:

Wir fügen dem bereits vorhandenen Programm

```
10 READ A,B
20 PRINT A,B,A+B,A*B,A/B
30 DATA 4, 5
```

folgende Zeilen an:

```
05 PRINT "BEISPIEL 6"
08 PRINT " A"," B","A+B","A*B","A/B"
```

Der Systembefehl RUN bewirkt, daß das Programm vor der Ausführung nach Zeilennummern geordnet wird.

Unser Programm ist dann in der folgenden Form gespeichert:

```
05 PRINT "BEISPIEL 6"
08 PRINT " A"," B","A+B","A*B","A/B"
10 READ A,B
20 PRINT A,B,A+B,A*B,A/B
30 DATA 4, 5
```

Wir erkennen nun, warum die Zeilen ursprünglich nicht mit fortlaufenden Nummern versehen wurden. Wir können nun Korrekturen leicht einfügen.

Wir erkennen auch die Bedeutung der Anführungszeichen in den PRINT-Anweisungen 5 und 8: Der zwischen den Anführungszeichen stehende Text wird ausgedruckt.

ÜBUNGEN

Für die folgenden Rechnungen sind Programme zu erstellen und der Output anzugeben[1].

1.01 Berechnung von $14 \cdot 0{,}5 \cdot 0{,}782$

1.02 Berechnung von $5 \cdot 3 - 6 \cdot 7 + 3 : 5$

$7 \cdot 8 - 16 \cdot 7 + 8 : 32$

1.03 Welche Ausgabe bewirken die Programme:

a)
```
10 READ X
20 PRINT 5*X-2
30 DATA 10
```

b)
```
10 READ X
20 PRINT 5*X, -2
30 DATA 10
```

c)
```
10 READ X, Y
20 PRINT X, 2*Y, X*Y,
30 PRINT Y*Y, Y*Y*Y
40 DATA 2, 3
```

d)
```
127 READ A, B
129 PRINT A, 2*B, A*B
131 PRINT B*B, B*B*B
133 DATA 2, 3
```

[1] output (engl.) ≙ Ausgabe

WEITERE BEISPIELE

7. Wir wollen Programm 5 von Seite 22 so erweitern, daß die Rechenoperationen $a+b$, $a \cdot b$ und $a:b$ auch für die folgenden geordneten Zahlenpaare (a, b) durchgeführt werden:

a	4	34	356.2	10	82.5
b	5	13	817.2	0	0.0567

Wir lösen diese Aufgabe mit Hilfe der Sprunganweisung **GOTO** n.

```
HONEYWELL BULL TIME-SHARING

ON AT   8:32   MARK I 26/02/75   TTY 50

USER NUMBER--W18170,TGMTGM
SYSTEM--BAS
NEW OR OLD--NEW BEI7
READY.

TAP                                  Umschaltung auf Lochstreifenbetrieb
READY                                Der Lochstreifenleser wird eingeschaltet
8  PRINT " A"," B","A+B","A*B","A/B"
10 READ A,B
20 PRINT A,B,A+B,A*B,A/B
25 GOTO 10
30 DATA 4, 5,  34, 13,  356.2, 817.2
40 DATA 10, 0,  82.5, .0567
RUN                                  Systembefehl für die Programmausführung


BEI7        8:34   MARK I 26/02/75

 A              B              A+B            A*B            A/B
 4              5              9              20             .8
 34             13             47             442            2.61538
 356.2          817.2          1173.4         291087.        .435879
 10             0              10             0

DIVISION BY ZERO  IN 20
                                                             5.78960E+76
 82.5           .0567          82.5567        4.67775        1455.03

OUT OF DATA  IN 10


USED     1.67 UNITS.                 Rechenzeit für das obige Programm
BYE                                  Systembefehl für die Trennung des Terminals vom Rechner

*** OFF AT  8:35   MARK I 26/02/75
     2 ELAPSED TERMINAL MINUTES      Gesamte Anschaltzeit
     1.67 TOTAL CRU'S USED           Gesamte Hauptrechnereinheiten
```

Die Sprunganweisung 25 GOTO 1Ø bewirkt, daß der Rechner nach der Ausführung der Anweisung 2Ø bei der Zeilennummer 1Ø weiterrechnet. Er belegt die Variablen A und B mit den nächsten Zahlen, also mit 34 und 13. Nach Ausführung der Anweisung 2Ø (Ausdrucken der Ergebnisse) bewirkt die nächste Anweisung, also 25 GOTO 1Ø, einen neuerlichen Rücksprung zu Zeile 1Ø usw.

Wenn der Rechner in der letzten DATA-Zeile keine Zahlen mehr vorfindet, mit denen er die Variablen A und B belegen kann, so meldet er OUT OF DATA IN 1Ø.

Beachte: Wir haben mit der Divisionsaufgabe A/B für das Zahlenpaar A=1Ø, B=Ø den Rechner zur Ausführung einer nicht erlaubten Rechenoperation, nämlich der Division durch Null, verleitet. Aus dem Ausgabeprotokoll ersehen wir, daß der Rechner in diesem Fall zunächst die Warnmeldung

DIVISION BY ZERO IN 20

gibt, für den undefinierten Quotienten den höchsten ihm zur Verfügung stehenden Wert, das ist 5.7896045E76, setzt, diesen laut Programm ausdruckt (6 geltende Ziffern!) und dann das Programm fortsetzt[1].

Ausdrücklich sei festgestellt:

Bei dem hier beschriebenen System werden zunächst das Programm und die Daten eingelesen.

Der Systembefehl RUN bewirkt die Ausführung des Programms.

Zunächst wird das im Arbeitsspeicher befindliche Programm nach aufsteigenden Zeilennummern geordnet.

Dann wird das Quellenprogramm[2] mit Hilfe des BASIC-Compilers im Hauptspeicher in ein Objektprogramm[3] umgewandelt.

Formale Fehler, also Verstöße gegen die Programmiersprache, werden am Terminal bekanntgegeben.

Alle Konstanten in den DATA-Anweisungen werden in einem Datenblock zusammengefaßt.

Durch eine READ-Anweisung mit n Variablen werden die Variablen mit den n nächsten, noch nicht aufgerufenen Konstanten des Datenblocks belegt.

Wenn durch eine READ-Anweisung mehr Konstanten aufgerufen werden, als im Datenblock noch vorhanden sind, so stoppt das Programm mit dem Output OUT OF DATA IN .....

[1] *Beachte:* 5.7896ØE + 76 steht für $5{,}78960 \cdot 10^{76}$

[2] also das BASIC-Programm

[3] Maschinenprogramm

8. Es ist ein Programm für das Ausdrucken folgender Tabelle zu schreiben:

x	x^2	x^3	$\sqrt{x}$	$\frac{1}{x}$
2.0	...	...	...	...
5.2	...	...	...	...
....	...	...	...	...

Programm:

```
10 PRINT "  X","X↑2","X↑3","X↑(1/2)","1/X"
20 READ X
30 PRINT X,X↑2,X↑3,SQR(X),1/X
40 GOTO 20
50 DATA 2, 5.2, 0, -4
```

Ausgabe:

```
 X             X↑2            X↑3            X↑(1/2)        1/X
 2              4              8              1.41421        .5
 5.2            27.04          140.608        2.28035        .192308
 0              0              0              0

DIVISION BY ZERO  IN 30
                                                             5.78960E+76
-4              16             -64

SQUARE ROOT OF NEGATIVE NUMBER  IN 30
                                              2              -.25

OUT OF DATA  IN 20
```

Der Warnausdruck

```
SQUARE ROOT OF NEGATIVE NUMBER  IN 30
```

zeigt uns, wie der Computer mit negativen Radikanden verfährt. Er schreibt die Warnmeldung, zieht die Wurzel aus dem absoluten Betrag des Radikanden und setzt die Rechnung fort.

9. Von einem Kreis sei der Radius $r = 10{,}0$ gegeben.

Es ist ein Programm für die Berechnung des Durchmessers d, des Umfangs u und des Flächeninhalts A zu erstellen.

Lösung:

```
10 READ R
20 PRINT "R=";R, "D=";2*R
30 PRINT "U=";2*3.14159*R, "A=";3.14159*R↑2
40 DATA 10
```

Ausgabe:

```
R= 10          D= 20
U= 62.8318     A= 314.159
```

Beachte: Der Strichpunkt nach "R=" (Zeile 20) bewirkt, daß der Wert für r, also die Zahl 10, nicht in die nächste Spalte gedruckt wird, sondern im Anschluß an R=

10. Gegeben seien die Winkel $\alpha_1 = 10°$, $\alpha_2 = 37{,}15°$, $\alpha_3 = 90°$, $\alpha_4 = 180°$.

Es ist ein Programm für die Erstellung einer Tabelle mit $\alpha°$, α^g, $\widehat{\alpha}$ zu schreiben[1].

Lösung:

Es gelten die Beziehungen[2]: $\alpha° : \alpha^g = 9 : 10$

$\widehat{\alpha} : \alpha° = \pi : 180$

also $\alpha^g = \frac{10}{9}\alpha°$ und $\widehat{\alpha} = \frac{\pi}{180} \cdot \alpha° = 0{,}0174532925 \cdot \alpha°$

Programm:

```
10 PRINT "ALTGRAD","NEUGRAD","BOGENMASS"
20 READ A
30 PRINT A,10/9*A,A*1.74532925E-2        ...[3]
40 GOTO 20
50 DATA 10, 37.15, 90, 180
```

---

[1] ° ist das Zeichen für Altgrad. ᵍ ist das Zeichen für Neugrad ($1^g = 1$ Gon). $\widehat{\alpha}$ ist das Zeichen für das Bogenmaß des Winkels α.

[2] 1 rechter Winkel $= 90° = 100^g$. Das Bogenmaß $\widehat{\alpha}$ ist gleich der Maßzahl der Länge des zum Mittenwinkel α gehörigen Bogens am Einheitskreis.

[3] *Beachte:* E−2 steht für 10^{-2}, und die Klammer ist nicht notwendig!

$\frac{a}{b} \cdot c$ wird in BASIC durch A*C/B oder durch A/B*C dargestellt!

Ausgabe:

```
ALTGRAD        NEUGRAD        BOGENMASS
 10             11.1111         .174533
 37.15          41.2778         .64839
 90             100.            1.5708
 180            200.            3.14159

OUT OF DATA  IN 20
```

11. Es ist ein Programm für die Berechnung von $c = \sqrt{a^2 + b^2}$ für die Zahlenpaare (3; 4), (1.25; 2.36), (9.52; 6.72) zu schreiben.

Lösung:

```
10 PRINT "  A","  B","  C"
20 READ A,B
30 PRINT A,B,SQR(A*A+B*B)                    ...[1]
40 GOTO 20
50 DATA 3,4, 1.25,2.36, 9.52,6.72
```

Wir geben dem Rechner unser Programm über den Lochstreifenleser des Terminals bekannt. Die Lochstreifeneingabe wird durch den Befehl TAPE (abgekürzt: TAP) aufgerufen.

Der Systembefehl RUN bewirkt die Ausführung des Programms.

Nach Beendigung der Rechnung erhalten wir folgende Ausgabe:

```
 A              B              C
 3              4              5          ...[2]
 1.25           2.36           2.6706
 9.52           6.72           11.6528

OUT OF DATA  IN 20
```

Bemerkungen[3]:

Jede Zeile ist mit einer Nummer versehen. Man wählt vorteilhaft 10, 20, 30, 40, ..., weil man so die Möglichkeit für das Einschieben von weiteren Zeilen hat.

Zu 10: Die Anweisung PRINT "..." bewirkt, daß der zwischen den Anführungszeichen stehende Text (Überschrift, Erläuterung, Anweisung für den Operator, ...) im Programmablauf an dieser Stelle ausgeschrieben wird.

---

[1] Statt A*A+B*B kann man auch A↑2+B↑2 schreiben.

[2] *Beachte:* Nullen hinter dem Komma werden nicht ausgedruckt, statt 3.000000 schreibt der Rechner kurz: 3

[3] Diese Bemerkungen dienen zur Wiederholung und Vertiefung des bisher durchgenommenen Stoffs.

Zu 40: Nachdem c für das Wertepaar (3, 4) mit 5 berechnet worden ist, bewirkt 40 GOTO 20, daß bei 20 weitergerechnet wird, daß dort die nächsten Werte gelesen und den Variablen A und B zugewiesen werden.

Zu 50: Zu READ-Anweisungen gehören DATA-Anweisungen. Beim System Mark I können bis zu 1280 numerische Konstanten in DATA-Anweisungen angegeben werden.

ÜBUNGEN

1.04 Das folgende Programm und das Ausgabeprotokoll sind zu ergänzen:

```
12  READ  .....
14  PRINT  I, J, J/I
16  GOTO  12
18  DATA  8, 2, 0, 7, 3, 9
```

```
TESTPROGRAMM
8     2     .25
```

1.05 Warum schreibt man in der Überschriftenzeile 10 des 8. Beispiels (Seite 27) nicht

X^2, X^3, $\sqrt{X}$, $\frac{1}{X}$ statt X ↑ 2, X ↑ 3, X ↑ (1/2), 1/X ?

1.06 Das Programm und das Ausgabeprotokoll sind zu ergänzen:

```
10  PRINT  "UEBUNG 1.06"
20  READ  X
30  PRINT  X, 1/X, SQR(X)
40  .....
50  DATA  1, —3, 0, 5
```

```
UEBUNG 1.06
 1          1          1
—3 .....  .....  .....
```

1.07 Es ist ein Programm für die Umrechnung von Neugraden in Altgrade zu erstellen.

1.08 Was bewirkt die Anweisung 120 GOTO 40 in einem Programm?

2. GRUNDELEMENTE VON BASIC

Lernziele

Die Zeichenarten des Fernschreibers angeben können.

Den Unterschied der Druckformen der Ziffer Null und des Buchstabens O angeben können.

Die Begriffe *Numerische Information* und *Alphamerische Information* erklären können.

Die Begriffe *Konstanten*, *Variablen*, *Numerische Konstanten*, *Ganzzahlen*, *Fixkommazahlen*, *Gleitkommazahlen* erklären können.

Die Form einer numerischen Konstanten bestimmen und Konstanten mit Formfehlern aussondern können.

Die Begriffe *Rechnergenauigkeit*, *Rechner-Unendlich*, *Ganzzahl-Unendlich* und *Rechner-Auflösungsvermögen* erklären können.

Den Aufbau gewöhnlicher numerischer und alphamerischer Variablen angeben können.

Die Regeln für die Belegung der Anfangswerte von Variablen durch den Compiler angeben können.

Die Operationszeichen und ihre Bedeutung angeben können.

Die Prioritäten bei der Durchführung von Rechenoperationen angeben können.

Den Aufbau der unbedingten Sprunganweisung angeben können.

Die Bedeutung der Zeilennummer in Sprunganweisungen angeben können.

Die Wirkung von unbedingten Sprunganweisungen angeben und fehlerhafte Anweisungen aussondern können.

Den Aufbau und die Wirkung der STOP-Anweisung und der END-Anweisung angeben können.

In den einführenden Beispielen wurde gezeigt, wie leichtverständlich und schnell erlernbar die problemorientierte Programmiersprache BASIC ist. Der aufmerksame Leser ist bereits jetzt imstande, einfachste Programme zu schreiben.

In diesem Abschnitt werden die gewonnenen Erkenntnisse vertieft und erweitert. Testbeispiele mit entsprechenden Hinweisen geben dem Leser die Möglichkeit, sein Wissen zu überprüfen und eventuelle Lücken zu schließen.

2.1. ZEICHEN

Die in BASIC verfügbaren Zeichen sind

a) Buchstaben

A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z

Beachte, daß es in BASIC keine Kleinbuchstaben gibt!

b) Ziffern

0 1 2 3 4 5 6 7 8 9 ...[1]

c) Sonderzeichen

\+ − * / ↑ , ; " . < > =
? ( ) # $ [ ] Leerstelle ...[2]

Beachte: Jede Zeile darf höchstens 72 Zeichen einschließlich der Leerstellen (engl.: blanks) enthalten.

Leerstellen innerhalb von Anweisungen (jedoch nicht in Zeilennummern[1]) werden vom Rechner ignoriert.

2.2. NUMERISCHE UND ALPHAMERISCHE INFORMATION

In BASIC teilt man die zu verarbeitende Information in 2 Gruppen ein:

1. *Numerische Information*

   Zahlen, mit denen gerechnet werden kann.

2. *Alphamerische Information*[3]

   Besteht aus Buchstaben, Ziffern, Sonderzeichen.

   Alphamerische Information kann gelesen, gespeichert und ausgedruckt werden. Mit ihr kann aber nicht gerechnet werden.

   Beachte: Zeilennummern sind eine alphamerische Information.

2.3. NUMERISCHE KONSTANTEN

In BASIC unterscheidet man:

Ganzzahlen,
Fixkommazahlen und
Gleitkommazahlen.

2.3.1. Ganzzahlen

Allgemeine Form einer Ganzzahl:

±ZiZiZiZiZiZi

Zum Beispiel: 0 1 23 −173

Größte Ganzzahl *(Ganzzahl-Unendlich)* in Mark I: 1073741823

[1] ***Beachte:*** Die Null wird durchgestrichen!

[2] Die Zeichen ! & % \ können zwar eingegeben, jedoch nicht gedruckt werden. Bei ihrer Übermittlung vom Terminal an den Rechner wird ! in . und & in + und % in # und \ in ‡ umgewandelt.

[3] Das Wort *alphamerisch* setzt sich aus alphabetisch und numerisch zusammen.

2.3.2. Fixkommazahlen

Fixkommazahlen enthalten einen Dezimalpunkt.
Allgemeine Form einer Fixkommazahl:

±ZiZiZi . ZiZiZiZiZiZi

Zum Beispiel: −153.87
173.
−Ø.ØØ728
.921

2.3.3. Gleitkommazahlen

Allgemeine Form einer Gleitkommazahl:

±Zi . ZiZiZiZiZiZiZiE ± ZiZi

Zum Beispiel: 1.5387E+02 oder 1.5387EØ2 oder 1.5387E2
oder 153.87EØ oder .15387E3
1E6 oder .1E7 oder 1ØØØØE2

Größte Gleitkommazahl *(Rechner-Unendlich)* in Mark I: 5.7896Ø449E76

Die kleinste, von Null verschiedene Zahl *(Auflösungsvermögen des Rechners)* in Mark I: 4.31808422E-78

Wichtig ist noch der Begriff der *Rechnergenauigkeit.*

Man versteht darunter die Anzahl von Stellen (wobei führende Nullen nicht mitgerechnet werden), die ein Rechner verarbeitet[1].

Bei Mark I beträgt die Rechnergenauigkeit 9 Stellen.

Beachte:

Statt des bei uns üblichen Dezimalkommas wird in BASIC ein Dezimalpunkt verwendet.

Die Null vor dem Dezimalpunkt muß nicht angeschrieben werden.

Nichtzulässige Schreibweisen sind:

3,	Komma statt Punkt!
14/3	gilt in BASIC als arithmetischer Ausdruck und darf daher nicht in einer DATA-Zeile aufscheinen.
PI	Diese Zahl muß durch einen Näherungswert, z. B. 3.14159 oder 3.14159265, dargestellt werden.
E	Auch diese Zahl muß durch einen Näherungswert, z. B. 2.71828183, dargestellt werden.

Als Zeichen für eine beliebige numerische Konstante werden wir das Symbol **K** verwenden.

[1] 3.2 hat 2 geltende (signifikante) Ziffern, 3 ist die signifikanteste Ziffer dieser Zahl.
3.21 hat 3 geltende Ziffern, 3 ist die signifikanteste Ziffer dieser Zahl.
0,081 hat 2 geltende Ziffern, 8 ist die signifikanteste Ziffer dieser Zahl.

Die Form der Ausgabe einer Konstanten hängt von ihrem Wert ab:

Eine Zahl x wird als *Ganzzahl* ausgegeben, wenn sie eine ganze Zahl ist und $|x| \leqq 1073741823$ gilt.

Eine nichtganze Zahl x wird als *Fixkommazahl* ausgegeben, wenn für sie gilt: $10^{-6} \leqq |x| < 10^{6}$

Alle anderen Zahlen werden als *Gleitkommazahlen*, also in der Form ±Zi.ZiZiZiZiZiE±ZiZi, ausgegeben.

ÜBUNGEN

2.01 Es ist die Zahlenform anzugeben:

a) −56782 ..... b) 317.67E−04 .....
c) 0. ..... d) Ø .....
e) .17324 ..... f) −8.19E17 .....

2.02 Es sind die Anzahl der signifikanten Ziffern und die signifikanteste Ziffer anzugeben:

a) 85.2 ..... b) −.003914 .....
c) −17589 ..... d) 0.06780 .....
e) .0729E2 ..... f) 3E-10 .....

2.03 Welche der folgenden Zahlendarstellungen sind in einer DATA-Zeile nicht erlaubt?

a) 0,912 b) 0.056 c) $-.412.10^{2}$ d) 5/3

2.4. NUMERISCHE VARIABLEN

In BASIC werden numerische Variablen durch
einen beliebigen Buchstaben
oder durch einen beliebigen Buchstaben gefolgt von einer Ziffer
bezeichnet[1].

Zulässige Schreibweisen:

A	AØ	A1	...	A9
B	BØ	B1		B9
.	..	..		..
.	..	..		..
Z	ZØ	Z1	...	Z9

Nichtzulässige Schreibweisen:

2A Das erste Zeichen muß ein Buchstabe sein!

CY Zwei und mehr Buchstaben zur Bezeichnung einer Variablen sind nicht erlaubt!

X14 Es wurde vereinbart, daß nur ein Buchstabe und eine Ziffer zur Variablenkennzeichnung zugelassen sind!

[1] Diese Symbole bezeichnet man als *gewöhnliche Variablen* im Gegensatz zu den *indizierten Variablen*, die im Abschnitt 9. behandelt werden.

Falls z. B. in der Programmzeile 50 eine fehlerhafte Variable steht, so veranlaßt der Compiler das Ausdrucken der Fehlermeldung:

ILLEGAL VARIABLE IN 50

Als Symbol für eine beliebige numerische BASIC-Variable werden wir das Zeichen **V** verwenden.

Jede numerische Variable **V** wird am Beginn der Rechnung mit dem Wert 0 belegt[1]. Sie kann im Verlauf der Rechnung mit beliebigen numerischen Konstanten belegt werden.

ÜBUNGEN

2.04 Welche Variablennamen sind nicht zulässig?

a) Y4	b) 5A	c) C 7	d) ADAM
e) EV	f) E5	g) A3	h) I8

2.5. ZEICHEN FÜR RECHENOPERATIONEN

Symbol	Operation
+	Addition
−	Subtraktion
*	Multiplikation
/	Division
↑	Potenzieren

BEISPIELE

1.	a^2	A↑2		
2.	b^3	B↑3		
3.	$\sqrt{a}$	A↑(1/2)	oder	A↑.5
4.	$a^{3/2}$	A↑(3/2)	oder	A↑1.5
5.	a^{-2}	A↑(−2)		
6.	a^{n-1}	A↑(N−1)		

7. Anhand eines kleinen Programms sei die Wirkung der Operation ↑ gezeigt.

```
10 PRINT 3↑2, 3↑(-2), 3↑0
20 PRINT 0↑3, 0↑0, (-2)↑3
```

Ausgabe:

```
 9            .111111          1
 0            1               -8
```

[1] Automatische Initialisierung mit 0. Es gibt auch Varianten von BASIC, für die das nicht gilt!

Beachte:

Im Betriebssystem Mark I wird A ↑ X

für A ≧ 0 für alle Werte von X,

für A < 0 für ganzzahlige Werte von X durchgeführt.

Es gibt auch BASIC-Compiler, die die Potenzierung nur bei positiver Basis durchführen.

2.6. REIHENFOLGE VON RECHENOPERATIONEN

Der Rechner führt innerhalb einer Anweisung die Rechnungen in folgender Reihenfolge durch:

1. Operationen innerhalb einer Klammer. Bei geschachtelten Klammern wird zunächst die innere Klammer berechnet.
2. Potenzierungen.
3. Multiplikationen und Divisionen.
4. Additionen und Subtraktionen.

Für aufeinanderfolgende gleichartige Operationen gilt:

Additionen, Subtraktionen, Multiplikationen und Divisionen werden von links nach rechts ausgeführt,

Potenzierungen werden von rechts nach links ausgeführt.

Operationen innerhalb einer Klammer werden von innen nach außen ausgeführt.

BEISPIELE

1.	A*B*C	1. A*B 2. Ergebnis 1*C
2.	A*B+C*D	1. A*B 2. C*D 3. Ergebnis 1 + Ergebnis 2
3.	A+B*D ↑ E	1. D ↑ E 2. Ergebnis 1*B 3. A + Ergebnis 2
4.	A+(B+C)	1. B+C 2. A + Ergebnis 1
5.	(A−B)*(A+B)	1. A−B 2. A+B 3. Ergebnis 1*Ergebnis 2

ÜBUNGEN

2.05 Welche Operation führt der Computer zuerst durch?

a) A+B/C

b) A+B−C*B+D↑2

c) A−B*(C−E/F)

2.7. ALPHAMERISCHE KONSTANTEN

Eine *alphamerische Konstante* besteht aus einer Folge von höchstens 15 BASIC-Zeichen[1], die von Anführungszeichen begrenzt sind.

Zum Beispiel: "A", "1", "MUELLER", "WERT 5", "A · B", " ", " ". ...[2]

2.8. ALPHAMERISCHE VARIABLEN

Damit man die Textausgabe auch durch das Programm steuern kann, hat man in BASIC auch alphamerische Variablen eingeführt.

Alphamerische Variablen werden durch einen Buchstaben, gefolgt von einem Dollarzeichen, dargestellt.

Zum Beispiel: A$, Z$

Jede alphamerische Variable wird am Beginn der Rechnung durch den Compiler mit dem Wert " ", also der Leerkonstanten, belegt.

Hinweis: Wie man mit alphamerischen Variablen arbeitet, wird im Folgeband BASIC-Praxis ausführlich gezeigt.

2.9. GOTO-ANWEISUNG

Die *unbedingte Sprunganweisung* lautet:

GOTO *n*

Diese Anweisung bewirkt, daß das Programm nicht mit der folgenden Anweisung, sondern mit der in der Zeile n stehenden Anweisung fortgesetzt wird[3].

Der Sprung kann sowohl zu einer höheren als auch zu einer niedrigeren Zeilennummer erfolgen.

---

[1] außer Anführungszeichen

[2] " " heißt Leerkonstante und hat keinen Inhalt.

[3] go to △ geh nach

Das Verzweigen zu Zeilen, die eine DIM-, REM- oder DATA-Anweisung enthalten, ist verboten! ...[1]

Unzulässige GOTO-Anweisungen sind z. B.:

100 GOTO	Die Zeilennummer fehlt!
150 G0T0 10	Richtig: GOTO 10
200 GOTO 357126	Nur 1- bis 5ziffrige Zeilennummern sind zulässig.
250 GO 35	Richtig: GOTO 35
300 GOTO 300	Bewirkt endlose Schleife.

2.10. STOP-ANWEISUNG

Die Anweisung

```
STOP
```

bewirkt die Beendigung des Objektprogramms.
Diese Anweisung wird u. a. zur Abschirmung von Programmteilen verwendet.

2.11. END-ANWEISUNG

Die Anweisung

```
END
```

zeigt dem Compiler das Ende des Programms an.

Diese Schlußanweisung muß in Mark I nicht angeschrieben werden; sie wird vom Compiler automatisch als letzte Anweisung gesetzt[2].

2.12. REM-ANWEISUNG[3]

Die Anweisung

```
REM
```

ermöglicht das Einfügen von erklärenden Texten in ein Programm.

BEISPIELE

```
100 REM BERECHNUNG DER UMSATZSTEUER
350 REM UNTERPROGRAMM ZINSENBERECHNUNG
```

[1] DIM-Anweisung siehe Seite 86.
[2] In Mark III muß die END-Anweisung in der Zeile mit der höchsten Zeilennummer stehen!
[3] remark ≙ Anmerkung.

3. TERME

Lernziele

Den Begriff *Term* erklären können.

Terme aus Zahlen programmieren können.

Terme aus Zahlen und Variablen programmieren können.

Wir bezeichnen Kombinationen aus BASIC-Variablen, BASIC-Zahlen, BASIC-Rechenzeichen und BASIC-Funktionszeichen als BASIC-Terme.

Wie bereits angegeben, sind in BASIC nur Großbuchstaben zugelassen. Aus Gründen der Übersichtlichkeit und um Verwechslungen zu vermeiden, bezeichnen wir mathematische Variablen mit kursiven Kleinbuchstaben.

Als Symbol für einen beliebigen BASIC-Term führen wir das Zeichen T ein.

Beachte: In der Algebra ist es üblich, den Punkt als Multiplikationszeichen zweier Variabler wegzulassen.

In BASIC muß das Operationszeichen * jedesmal bei einer Multiplikation gesetzt werden!

Klammern haben in BASIC die in der Algebra übliche Bedeutung[1].

BEISPIELE

1.	$x-y$	X−Y
2.	$a-2b+3c$	A−2*B+3*C
3.	$ab-4cd+3xy$	A*B−4*C*D+3*X*Y
4.	$\frac{a}{x}-2\frac{b}{y}+3\frac{cd}{u}$	A/X−2*B/Y+3*C*D/U

[1] Außer beim Index und bei Funktionsargumenten!

5.	$\frac{a}{b}-c$	A/B−C
6.	$\frac{a}{b-c}$	A/(B−C)
7.	$\frac{a}{bc}$	A/B/C oder A/(B∗C)
8.	$\frac{a-b}{c+d}$	(A−B)/(C+D)
9.	ax^4-2bx^3+cx+d	A∗X↑4−2∗B∗X↑3+C∗X+D
10.	$(ax)^3$	(A∗X)↑3
11.	$\sqrt{a^2+b^2}$	(A∗A+B∗B)↑(1/2)
12.	$\sqrt[3]{\frac{a+b}{c(d-e)^2}}$	((A+B)/(C∗(D−E)↑2))↑(1/3)

Beachte: Jede Klammer besteht aus zwei Teilen, z. B. aus (und).

Dem algebraischen Term $(x+(y+z)^3)^2$ entspricht der BASIC-Term (X+(Y+Z)↑3)↑2

Beachte: Wenn mehrere Klammern ineinandergeschachtelt sind, so beginnt der Computer mit der Berechnung des durch die innerste Klammer gekennzeichneten Terms.

Klammern, die die Bedeutung eines Terms nicht verändern, sind erlaubt[1].

ÜBUNGEN

Die folgenden Terme sind in die BASIC-Sprache zu übersetzen.

3.01 $u-v+w$

3.02 $x-yz$

3.03 $ax+by-c$

3.04 $\frac{r}{s}-\frac{t}{u}$

[1] Statt (A/B)+(C−D)∗E kann man auch kürzer A/B+(C−D)∗E schreiben.

3.05 $\dfrac{x-3y}{z}$ .....

3.06 $\dfrac{d-4ef}{2gh} - a^2$

3.07 $(a-b)^3$ .....

3.08 $a-b^3$ .....

3.09 $\sqrt[3]{(a-b)^2}$ .....

3.10 $(a-2b)(2a+3b)$ .....

3.11 $a^{r-s} - b^{2r+3s}$ .....

3.12 $a^{r-s} \cdot b^{2r+s}$ .....

3.13 $\left(\dfrac{a}{2b}\right)^{-4}$ .....

3.14 $[a:(-b)]^{-2}$ .....

3.15 $\dfrac{a\sin\beta}{\sin\alpha}$ .....

3.16 $\sin^2\alpha$

3.17 $2\sin\dfrac{\alpha+\beta}{2}\cos\dfrac{\alpha-\beta}{2}$ .....

Die einwandfreie Beherrschung der BASIC-Schreibweise ist eine notwendige Voraussetzung für das Verständnis des weiteren Stoffs. Sollten Ihnen bei diesen Übungen Fehler unterlaufen sein, so liegt es in Ihrem Interesse, die Abschnitte 2 und 3 noch einmal gründlich durchzuarbeiten.

4. LET-ANWEISUNG (ERGIBT-ANWEISUNG)

Lernziele

Den Aufbau und die Wirkung der Ergibt-Anweisung erklären können.

Die Berechnung einer Formel und die Ausgabe des ermittelten Werts programmieren können.

Zulässige von unzulässigen Ergibt-Anweisungen unterscheiden können.

Für die Anweisungen

Berechne den Wert des Terms T

und Speichere diese Zahl unter dem Namen V

schreibt man in BASIC:

LET V=T	oder kurz:	V=T

Anders ausgedrückt: Durch die Anweisung V=T wird der Termwert von T berechnet und der Variablen V zugeordnet[1].

BEISPIELE

1. 100 LET P=3.14159 oder[2] P=3.14159

   bewirkt, daß die Variable P mit der Zahl 3.14159 belegt wird. Wenn im Verlauf der weiteren Rechnung 3.14159 benötigt wird, so kann man diese Zahl unter dem Namen P aufrufen.

2. 120 LET B=2+5 oder 120 B=2+5

   bewirkt, daß die Variable B mit der Zahl 7 belegt wird. Es wird also die Zahl 7 der Variablen B zugeordnet.

3. 240 S=0

   Durch diese Anweisung wird der Speicherplatz S auf Null gesetzt, d. h. S wird mit der Zahl Null belegt.

---

[1] Sonderfall: T kann auch eine Konstante sein.

[2] In diesem Buch wird oft vor und hinter dem Gleichheitszeichen zur Erhöhung der Übersichtlichkeit je eine Leerstelle geschrieben. In der Praxis schreibt man aus Kostengründen kürzer: P=3.14159

4. Die Variable A sei mit 4 belegt, die Variable B mit 3.

```
18Ø  X=A+B
```

bewirkt, daß die Variable X mit der Zahl 7 belegt wird, daß also die Zahl 7 unter dem Namen X gespeichert wird.

```
21Ø  Y=SQR(A↑2+B↑2)   im selben Programm
```

bewirkt, daß der Termwert der Quadratwurzel von A↑2+B↑2 für A=4 und B=3, also die Zahl 5 unter dem Namen Y gespeichert wird[1].

```
23Ø  Z=X*Y   im selben Programm
```

bewirkt, daß das Produkt des Inhalts des Speicherplatzes X (hier die Zahl 7) und des Inhalts des Speicherplatzes Y (die Zahl 5) unter dem Namen Z gespeichert wird.

```
24Ø  PRINT  X, Y, Z, Y↑2
```

im selben Programm bewirkt, daß die Zahlen 7 5 35 25 gedruckt werden.

Beachte, daß die Anweisung `22Ø  PRINT  X, Z, Y`

die Ausgabe 7 Ø 5 bewirkt.

Z ist noch nicht mit der Zahl 35 belegt (erst in der Zeile 23Ø).

Nochmals:

> In BASIC wird jeder Variablen am Beginn der Rechnung die Zahl Null zugeordnet!

5.
```
54Ø  A=1
.....
6ØØ  A=5
.....
7ØØ  PRINT  A↑2
```

Die Anweisung 7ØØ bewirkt hier das Ausdrucken der Zahl 25, die Variable A ist also mit der Zahl 5 belegt.

Beachte: Durch 54Ø ... wurde A mit 1 belegt.

Durch 6ØØ ... wird A mit 5 belegt. Bei dieser Speicherung wird der „alte" Wert von A, die Zahl 1, gelöscht.

[1] Die Zahl 5 ist unter dem Namen Y in der weiteren Programmfolge so lange aufrufbar, bis Y mit einer anderen Zahl belegt wird.

6. Die *Mehrfachzuweisung*

```
110 D1=D2=D3=B↑2-4*A*C
```

bewirkt, daß der Wert des Terms B↑2−4*A*C unter den Namen D1 und D2 und D3 gespeichert wird.

7.
```
170 X=4
-----------------------
200 X=X+1
```

Die Anweisung 200 bewirkt, daß der Termwert von X+1, also die Zahl 5, unter dem Namen X gespeichert wird.

Beachte: Der BASIC-Anweisung 200 X=X+1 entspricht die Zuordnung: $x \leftarrow x+1$.

Die Sprechweise lautet: $x+1$ ergibt x und wurde vor fast 30 Jahren von Konrad Zuse eingeführt[1].

Beachte die verschiedenen Bedeutungen	
der BASIC-Anweisung	X=X+1
und der algebraischen Gleichung	$x=x+1$ ...[2]

8. Die Anweisungen

```
150 A = B = C = 2
160 B = A-3*C
170 PRINT A,B,C
```

bewirken folgendes Protokoll:

```
 2            -4            2
```

9. Viele Aufgaben sollen gleich für einen Satz von Argumenten gerechnet werden, wobei mit einem Anfangswert x_0 begonnen wird und mit fester Schrittweite D fortgesetzt wird.

Für $x_1 = x_0 + D$
$x_2 = x_1 + D$
$x_3 = x_2 + D$ usw.

schreibt man in der Mathematik kurz: $x_{i+1} = x_i + D$

Die analoge BASIC-Anweisung lautet:

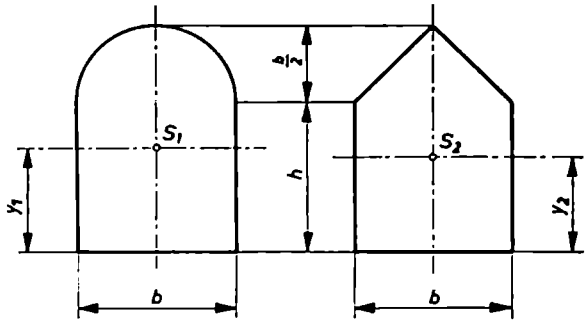
X=X+D

---

[1] K. Zuse verwendete anstatt des Zeichens ← den Doppelpfeil ⇐.

[2] Für die Anweisung X = X + 1 schreibt man in der Mathematik oft: $x_{neu} = x_{alt} + 1$

10. Es sind die Flächeninhalte, die Trägheitsmomente und die Widerstandsmomente der folgenden Figuren zu ermitteln:



Lösung: Die benötigten Formeln lauten:

$$A_1 = b \cdot h + \frac{b^2 \cdot \pi}{8}$$

$$A_2 = b \cdot h + \frac{b^2}{4}$$

$$y_0 = \frac{2b}{3\pi}$$

$$y_1 = \frac{b \cdot h^2 + \frac{b^2 \pi}{4}(h + y_0)}{2A_1}$$

$$y_2 = \frac{12 \cdot b \cdot h^2 + 6b^2 h + b^3}{24 \cdot A_2}$$

$$I_1 = \frac{b \cdot h^3}{12} + b \cdot h\left(y_1 - \frac{h}{2}\right)^2 + 0.1098\left(\frac{b}{2}\right)^4 + \frac{b^2 \cdot \pi}{8}(h + y_0 - y_1)^2$$

$$I_2 = \frac{b \cdot h^3}{12} + b \cdot h\left(y_2 - \frac{h}{2}\right)^2 + \frac{b}{36}\left(\frac{b}{2}\right)^3 + \frac{b^2}{4}\left(h + \frac{b}{6} - y_2\right)^2$$

$$W_{1\,\text{oben}} = W_1 = \frac{I_1}{h + \frac{b}{2} - y_1} \qquad W_{1\,\text{unten}} = V_1 = \frac{I_1}{y_1}$$

$$W_{2\,\text{oben}} = W_2 = \frac{I_2}{h + \frac{b}{2} - y_2} \qquad W_{2\,\text{unten}} = V_2 = \frac{I_2}{y_2}$$

Programm:

```
110 P = 3.141592654
120 K = 2/(3*P)
130 READ B,H
140 A1 = B*H+P*B↑2/8
150 A2 = B*H+B↑2/4
160 Y0 = K*B
170 Y1 = (B*H↑2+P/4*B↑2*(H+Y0))/(2*A1)
180 Y2 = (12*B*H↑2+6*B↑2*H+B↑3)/(24*A2)
190 I1 = B*H↑3/12+B*H*(Y1-H/2)↑2
200 I1 = I1+.1098*(B/2)↑4+P/8*(B*(H+Y0-Y1))↑2
210 I2 = B*H↑3/12+B*H*(Y2-H/2)↑2+B↑4/(8*36)
220 I2 = I2+(B*(H+B/6-Y2)/2)↑2
230 W1 = I1/(H+B/2-Y1)
240 V1 = I1/Y1
250 W2 = I2/(H+B/2-Y2)
260 V2 = I2/Y2
270 PRINT "B =";B,"A1 =";A1,"A2 =";A2
280 PRINT "H =";H,"Y1 =";Y1,"Y2 =";Y2
290 PRINT " ","I1 =";I1,"I2 =";I2
300 PRINT " ","W1 =";W1,"W2 =";W2
310 PRINT " ","V1 =";V1,"V2 =";V2
320 PRINT
330 PRINT
340 GOTO 130
350 DATA 4, 3, 5.17, 6.83
```

Ausgabe:

```
B =  4          A1 =  18.2832    A2 =  16
H =  3          Y1 =  2.3072     Y2 =  2.04167
                I1 =  33.5084    I2 =  23.9722
                W1 =  12.4437    W2 =  8.10329
                V1 =  14.5234    V2 =  11.7415


B =  5.17       A1 =  45.8075    A2 =  41.9933
H =  6.83       Y1 =  4.44891    Y2 =  4.09553
                I1 =  306.902    I2 =  242.519
                W1 =  61.7996    W2 =  45.5907
                V1 =  68.9836    V2 =  59.2154


OUT OF DATA  IN 130
```

ÜBUNGEN

Übertrage die folgenden Wertzuordnungen in BASIC und gib jeweils den unter dem Variablennamen gespeicherten Wert an, wenn $a=1$, $b=2$, $c=3$, $d=4$, $q=-3$ und $\alpha=30°$ angenommen wird und *4.01*, *4.02*, ... aufeinanderfolgende Zeilen **eines** Programms sind.

4.01 $x=3a-2b+c$ 4ϕ1 X = ...

4.02 $y=5a-\frac{d}{c}$ 4ϕ2

4.03 $z=\sqrt{a^2+b^2+c^2+d^2}$ 4ϕ3

4.04 $p_1=\frac{ax+b}{cx+d}$ 4ϕ4

4.05 $p_2=\frac{ax^2+bx+c}{bx^2-ax+d}$ 4ϕ5

4.06 $s_3=1+q+q^2+q^3$ 4ϕ6

4.07 $c=\frac{3{,}14159}{180}\cdot\alpha$ 4ϕ7 .....

4.08 $x_1=\frac{-b+\sqrt{b^2-4ac}}{2a}$ 4ϕ8 .....

5. FUNKTIONEN

Lernziele

Den Aufbau einer Standardfunktion angeben können.

Die Wirkung der Standardfunktionen

SQR ABS SGN LOG EXP
SIN COS TAN ATN INT RND

erklären können.

Die Umrechnung von Altgraden und von Neugraden in das Bogenmaß programmieren können.

Zehnerlogarithmen programmieren können.

Das Runden von Zahlen programmieren können.

Zufallszahlen innerhalb eines vorgegebenen Bereichs programmieren können.

Den Aufbau einer DEF-Anweisung angeben können.

Die Wirkung einer DEF-Anweisung erklären können.

In BASIC unterscheiden wir zwischen

Standardfunktionen und
vereinbarten Funktionen.

5.1. STANDARDFUNKTIONEN[1]

Wie wir bereits sehen konnten, ist es in BASIC möglich, einige Funktionen durch einfache Symbole aufzurufen.

Aufbau einer Standardfunktion:

Name (Argument)

Der Funktionsname besteht aus drei Buchstaben.

Das in Klammern gesetzte Argument kann ein beliebiger BASIC-Term sein, also eine sinnvolle Verknüpfung von numerischen Konstanten, numerischen Variablen und Funktionen.

Die Berechnung eines Funktionswerts hat Vorrang vor allen übrigen Rechenoperationen.

T ist ein beliebiger BASIC-Term und t sei dessen Wert.

[1] Eingebaute Funktionen (built-in functions).

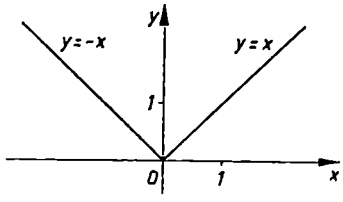
5.1.1. Quadratwurzelfunktion SQR

SQR(T) bedeutet $\sqrt{t}$ für $t \geqq 0$.

5.1.2. Absolutwertfunktion ABS

ABS(T) bedeutet $|t|$.

Der Absolutbetrag $|t|$ ist definiert durch:



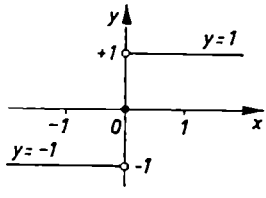
$$|t| = \begin{cases} t & \text{für } t > 0 \\ 0 & \text{für } t = 0 \\ -t & \text{für } t < 0 \end{cases}$$

Zum Beispiel: ABS(9.8)=9.8 ABS(Ø)=Ø ABS(−9.8)=−(−9.8)=9.8

5.1.3. Vorzeichenfunktion SGN

SGN(T) bedeutet sgn(t).

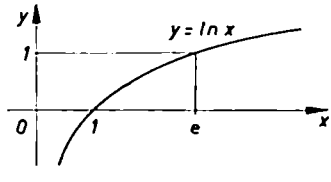
Die Signumfunktion (Vorzeichenfunktion) ist folgendermaßen definiert:



$$\operatorname{sgn}(t) = \begin{cases} +1 & \text{für } t > 0 \\ 0 & \text{für } t = 0 \\ -1 & \text{für } t < 0 \end{cases}$$

5.1.4. Logarithmusfunktion LOG

LOG(T) bedeutet $\ln t$ für $t > 0$.



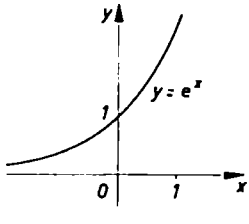
Beachte:

LOG(X)/LOG(10),
also .434294482*LOG(X) erzeugt $\lg x$ für $x > 0$.

LOG(X)/LOG(2),
also 1.44269504*LOG(X) erzeugt $\log_2 x$ für $x > 0$.

5.1.5. Exponentialfunktion EXP

EXP(T) bedeutet e^t, auch $\exp(t)$ geschrieben, wobei $e = 2.71828183$ die Basis des natürlichen Logarithmus ist.



5.1.6. Kreisfunktionen SIN, COS, TAN

SIN(T) bedeutet $\sin t$
COS(T) bedeutet $\cos t$
TAN(T) bedeutet $\tan t$
} wobei t als Bogenmaß aufgefaßt wird.

Beachte:

SIN(X*1.74532925E−2)
bzw.
SIN(X/57.2957795)
} erzeugt $\sin x$, wobei x als Gradmaß aufgefaßt wird.

BEISPIELE

1. Durch die Anweisung

   150 Y=SIN(.5)

   wird der Sinuswert des Winkels, dessen Bogenmaß 0,5 beträgt ($\triangleq$28,64789°), berechnet und unter dem Namen Y gespeichert.[1]

2. Durch die Anweisungen

   72 X=.3

   74 Y=.6

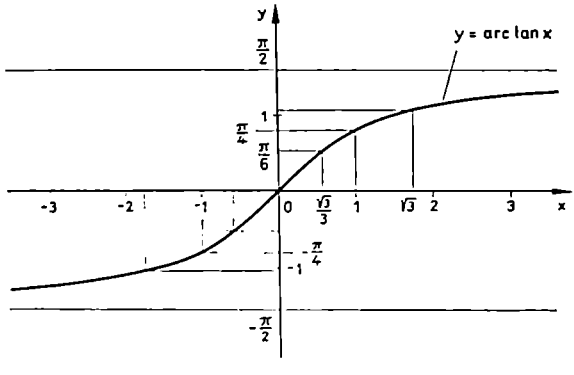
   75 Z=COS(SQR(X*X+2.1*Y))

   wird

   1. die Variable X mit 0,3 belegt,
   2. die Variable Y mit 0,6 belegt,
   3. der Wert des Rechtsterms der Zeile 75 berechnet,
   4. dieser Wert (0.397601) unter dem Namen Z gespeichert.

5.1.7. Arkustangensfunktion ATN

ATN(T) bedeutet arctan$t$. ...[2]



arctan$x$ ist jener Bogen zwischen $-\frac{\pi}{2}$ und $+\frac{\pi}{2}$, dessen Tangens den Wert x hat.

[1] sin 0,5 = 0,479425539

[2] ATN(T) wird gelesen: Arkustangens T
100 B=ATN(X) bewirkt, daß der Rechner die Variable B mit einer Zahl, deren Absolutbetrag kleiner oder gleich 1,57079633 $\left(\text{also } \frac{\pi}{2}\right)$ ist, belegt.

BEISPIELE

1. $\text{arctan}\,0 = 0 \qquad \arctan 1 = \frac{\pi}{4} \qquad \arctan(-\sqrt{3}) = -\frac{\pi}{3}$

ATN(1ØØØ) = 1.5698 $\triangleq$ 89.9427°

ATN(1E6) = 1.57Ø8 $\triangleq$ 89.9999°

2. Es ist mit Hilfe der Arkustangensfunktion der Wert von π zu berechnen.

Es gilt: $\tan\frac{\pi}{4} = 1$, also $\frac{\pi}{4} = \arctan 1$ und somit $\pi = 4 \cdot \arctan 1$

Programm:

```
1Ø PRINT "PI =";4*ATN(1)
```

Ausgabe:

```
PI = 3.14159
```

3. Von einem rechtwinkligen Dreieck sind $a = 9{,}52$ cm und $b = 6{,}72$ cm gegeben. Es ist ein Programm für die Berechnung der Seite c und der Winkel α und β zu schreiben.

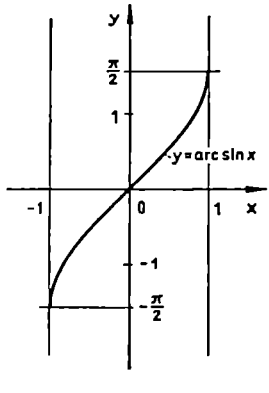
Programm:

```
1Ø READ A,B
2Ø W1 = ATN(A/B)/1.74532925E-2
3Ø PRINT "A =";A,"B =";B,"C =";SQR(A↑2+B↑2)
4Ø PRINT "ALPHA =";W1,"BETA =";9Ø-W1
5Ø DATA 9.52, 6.72
```

Ausgabe:

```
A = 9.52          B = 6.72          C = 11.6528
ALPHA = 54.7824                     BETA = 35.2176
```

4. Es ist ein Sinuswert x gegeben. Der zugehörige Winkel y ist zu berechnen.



Lösung: $y = \arcsin x$

$\arcsin x$ ist jener Bogen zwischen $-\frac{\pi}{2}$ und $+\frac{\pi}{2}$, dessen Sinus den Wert x hat.

In BASIC steht uns die Funktion $\arcsin x$ nicht zur Verfügung. Wir müssen daher $\arcsin x$ durch $\arctan x$ ausdrücken.

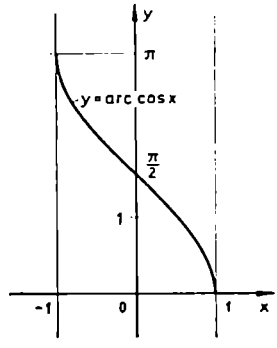
Aus $\tan y = \dfrac{\sin y}{\cos y} = \dfrac{\sin y}{\sqrt{1-\sin^2 y}} = \dfrac{x}{\sqrt{1-x^2}}$

folgt $y = \arctan \dfrac{x}{\sqrt{1-x^2}}$ für $|x| < 1$

$$\arcsin x = \begin{cases} \arctan \dfrac{x}{\sqrt{1-x^2}} & \text{für } |x| < 1 \\ \dfrac{\pi}{2} & \text{für } x = 1 \\ -\dfrac{\pi}{2} & \text{für } x = -1 \end{cases}$$

Wie man diese Funktion programmiert, wird im Beispiel 2 auf Seite 115 gezeigt.

5. Es ist ein Kosinuswert x gegeben. Der zugehörige Winkel y ist zu berechnen.



Lösung: $y = \arccos x$

$\arccos x$ ist jener Bogen zwischen 0 und π, dessen Kosinus den Wert x hat.

Weil in BASIC nur ATN aufgerufen werden kann, müssen wir $\arccos x$ durch $\arctan x$ ausdrücken.

Aus $\tan y = \dfrac{\sin y}{\cos y} = \dfrac{\sqrt{1-\cos^2 y}}{\cos y} = \dfrac{\sqrt{1-x^2}}{x}$

folgt $y = \arctan \dfrac{\sqrt{1-x^2}}{x}$ für $x \neq 0$

Bei der Umrechnung in arctan müssen wir berücksichtigen, daß durch ATN ein Bogen größer als $-\dfrac{\pi}{2}$ und kleiner als $\dfrac{\pi}{2}$ ermittelt wird, jedoch durch $\arccos x$ ein Bogen $\geqq 0$ und $\leqq \pi$.

$$\arccos x = \begin{cases} \arctan \dfrac{\sqrt{1-x^2}}{x} & \text{für} \quad 0 < x \leqq 1 \\ \dfrac{\pi}{2} & \text{für} \quad x = 0 \\ \pi + \arctan \dfrac{\sqrt{1-x^2}}{x} & \text{für} \quad -1 \leqq x < 0 \end{cases}$$

Die Programmierung dieser Funktion wird bei der Behandlung der kubischen Gleichung auf Seite 122 gezeigt.

5.1.8. Integerfunktion INT ...[1]

INT(T) erzeugt die größte ganze Zahl, die nicht größer als der Wert von T ist.

Zum Beispiel: INT(4) = 4 INT(8.9) = 8 INT(−2.12) = −3

Mit Hilfe der Integerfunktion kann man eine Zahl x folgendermaßen auf r Dezimalstellen runden[2]:

$\dfrac{\text{INT}(10^r \cdot x + 0{,}5)}{10^r}$ für positive x

$\text{sgn}(x) \cdot \dfrac{\text{INT}(10 \cdot |x| + 0{,}5)}{10}$ für reelle x

BEISPIELE

1. Die Zahl 7,2587 ist auf eine ganze Zahl und dann auf 1, 2 und 3 Dezimalstellen zu runden.

INT(7.2587+.5) = 7

INT(10*7.2587+.5)/10 = INT(72.587+.5)/10
= INT(73.087)/10 = 73/10 = 7.3

INT(100*7.2587+.5)/100 = INT(725.87+.5)/100
= INT(726.37)/100 = 726/100 = 7.26

INT(1000*7.2587+.5)/1000 = INT(7259.2)/1000 = 7259/1000 = 7.259

2. Die Zahl −2,15 ist auf Zehntel zu runden.

SGN(−2.15)*INT(10*ABS(−2.15)+.5)/10
= −INT(21.5+.5)/10 = −2.2

[1] Integer (gespr.: 'intid3ə) ≙ Ganzzahl, ganzzahlig

5.1.9. Randomfunktion RND ...[1]

Aus einer Tabelle von 2^{30} Zufallszahlen, die zwischen 0 und (ausschließlich) 1 liegen, erzeugt RND(T) für

a) $t>0$ eine ganz bestimmte Zufallszahl,

b) $t<0$ eine von der Uhrzeit abhängige Zufallszahl,

c) $t=0$ die jeweils nächste Zufallszahl der Tabelle.

Eine Folge von 10 Zufallszahlen kann man auf folgende Weise erzeugen:

```
10 X = RND(1)
20 FOR I = 1 TO 10
30 PRINT RND(0)
40 NEXT I
```

Ersetzt man die Zeile 10 durch 10 X=RND(−1)

so werden bei jedem Lauf dieses Programms andere Zahlen erzeugt!

Beachte: Die Zufallszahlen spielen in der Stichprobentheorie und damit in der Qualitätsprüfung eine große Rolle.

BEISPIELE

1. Das Programm

```
10 FOR I = 1 TO 3
20 PRINT RND(0),RND(1),RND(1000),RND(-5),RND(-1)
30 NEXT I
```

bewirkte beim 1. Lauf z. B. die Ausgabe

```
 .709245      .98633       .462892      .901821      7.24023E-02
 .628688      .98633       .462892      .901821      7.24023E-02
 .628688      .98633       .462892      .903775      7.63085E-02
```

und beim folgenden Lauf z. B. die Ausgabe:

```
 .709245      .98633       .462892      .429165      .12709
 .683375      .98633       .462892      .429165      .12709
 .683375      .98633       .462892      .431118      .130996
```

2. Es sind Programme für die Erzeugung von 20 Zufallszahlen zu erstellen. Diese Zahlen sollen

a) 6 Dezimalstellen besitzen;

b) einstellige ganze Zahlen sein;

c) der Menge {2, 3, 4, 5, 6} angehören;

d) ganze Zahlen größer oder gleich 17 und kleiner als 100 sein;

[1] random ≙ zufällig
In diesem Abschnitt wird die Kenntnis der FOR-Schleife (Abschnitt 8.2.) vorausgesetzt!

a) Programm:

```
10 Z = RND(1)
20 FOR I = 1 TO 20
30 PRINT RND(0),
40 NEXT I
```

Beachte: Jedes positive Argument von RND(T) leitet eine ganz bestimmte Zufallszahlenfolge ein.

Ausgabe:

```
 .473599        .442519        .498805        .373168        .921321
 .123978        8.68505E-02    4.06526E-02    .341097        .468896
2.97623E-02     .75441         .498551        .242898        9.31652E-02
 .280064        .159309        .211611        .042684        .383241
```

b) Wenn die Zufallszahlen einstellige ganze Zahlen sein sollen, so wird die Zeile 30 des ersten Programms ersetzt durch

```
30 PRINT INT(10*RND(0)),
```

Die Ausgabe hat dann folgende Form:

```
 4              4              4              3              9
 1              0              0              3              4
 3              7              4              2              0
 2              1              2              0              3
```

Beachte: $0 \leqq \mathrm{RND(T)} < 1$

$0 \leqq 10 * \mathrm{RND(T)} < 10$

c) Wenn die Zufallszahlen zwischen 2 und 6 (einschließlich) liegen sollen, so lautet die Zeile 30:

```
30 PRINT INT(5*RND(0)+2),
```

Ausgabe:

```
 4              4              4              3              6
 2              2              2              3              4
 2              5              4              3              2
 3              2              3              2              3
```

Beachte: $0 \leqq \mathrm{RND(T)} < 1$

$0 \leqq 5 * \mathrm{RND(T)} < 5$

$2 \leqq 5 * \mathrm{RND(T)} + 2 < 7$ $\qquad N \leqq M * \mathrm{RND(T)} + N < M + N$

Allgemein gilt:

$$Z = A + (E - A) * \mathrm{RND(T)}$$

erzeugt eine Zufallszahl, die zwischen A und (ausschließlich) E liegt.

d) Die Zeile 30 des ersten Programms muß, wenn die Zufallszahlen größer oder gleich 17 und kleiner als 100 sein sollen, geändert werden auf ...[1]

```
30 PRINT INT(83*RND(0)+17),
```

Ausgabe:

```
56          53          58          47          93
27          24          20          45          55
19          79          58          37          24
40          30          34          20          48
```

Beachte:

In Mark III-BASIC verwendet man

- statt RND(0) die Funktion RND
  RND erzeugt eine in]0, 1[liegende Zufallszahl.
- statt RND(−1) die Funktion RAN
  RAN startet den Zufallsgenerator mit einer Zufallszahl.

ÜBUNGEN

5.01 Bei welchen Funktionen muß das Argument positiv sein?

5.02 Bei welchen Funktionen darf das Argument nicht negativ sein?

5.03 Welchen Wert kann SQR(X) höchstens annehmen?

5.04 Der Definitions- und der Wertebereich von LOG(X) sind anzugeben.

5.05 Der Definitions- und der Wertebereich von EXP(X) sind anzugeben.

5.06 Der Funktionswert ist anzugeben:

a) ABS(INT(−6.8)) .....

b) INT(ABS(−6.8)) .....

c) EXP(COS(0)) .....

d) ATN(TAN(1.4)) .....

e) ATN(1.)

[1] N=17 ∧ M+N=100 ⇒ M=83

Es ist zu programmieren:

5.07 sin 12° 17′ 36″

5.08 lg 132

5.09 arctan(3.2) in Altgraden

5.10 arctan(3.2) in Neugraden

5.11 Ganze Zufallszahl zwischen 13 und 78

5.12 Zufallszahlen zwischen −9,8 und 5,3

5.13 Rundung einer positiven Zahl x auf Hundertstel

5.14 Rundung einer beliebigen Zahl x auf Hundertstel

5.2. VEREINBARTE FUNKTIONEN

Neben den Standardfunktionen können in BASIC beliebige Anweisungsfunktionen definiert werden. Die entsprechende Anweisung lautet:

```
DEFFNA(X)=T
```

DEF ist eine Abkürzung für Definition.

Der Funktionsname FN*b* besteht aus drei Buchstaben, nämlich FN an erster und zweiter Stelle und einem beliebig zu wählenden an dritter Stelle.

Das Argument X ist eine Variable.

T ist ein beliebiger, in einer Zeile darstellbarer BASIC-Term[1].

Eine durch eine DEF-Anweisung definierte Funktion ist, wie die Standardfunktionen, an jeder Stelle im Programm, also auch vor der DEF-Anweisung, aufrufbar.

BEISPIELE

1. Programm:

```
100 DEF FNA(X) = X*X+1
110 Z = 2
120 K5 = FNA(Z+3)
130 PRINT K5, FNA(10), FNA(Z↑2-1)
```

Ausgabe:

```
 26            101            10
```

[1] T darf nicht die links vom Gleichheitszeichen stehende Funktion enthalten; es ist somit .. DEF FNA(X)=FNA(Y)+1 nicht erlaubt.
Ebenso ist .. DEF FNA(X)=FNB(X) nicht erlaubt!

In der Mathematik sind $x \to x^2+1$ und $y \to y^2+1$ zwei Darstellungen derselben Funktion.

Analog gilt in BASIC:

```
FNA(X)=X*X+1   und
FNA(Y)=Y*Y+1
```

sind verschiedene Darstellungen der Funktion FNA.

Die Variable einer DEF-Anweisung ist nur für die Rechenvorschrift innerhalb der DEF-Anweisung von Bedeutung. Sie hat nichts mit einer gleichnamigen Variablen im übrigen Programm zu tun.

Man spricht daher von einer *Scheinvariablen* oder einem *Formalparameter*.

Zu Zeile 120: Durch die Anweisung K5=FNA(Z+3)
wird statt des Formalparameters X der momentane Wert von Z, also 2, eingesetzt. Dann wird FNA für das Argument Z+3, also für 5, berechnet. Sodann wird der Termwert 26 an die Variable K5 weitergegeben.

Zu Zeile 130:

Durch FNA(10) wird die in Zeile 100 definierte Funktion FNA aufgerufen, der Formalparameter X durch die Zahl 10 ersetzt und der Termwert 10^2+1 berechnet.

Da diese Anweisung in einer PRINT-Zeile steht, wird der Termwert 101 gedruckt.

Durch FNA(Z↑2−1) wird die in Zeile 100 definierte Funktion FNA aufgerufen. Der Formalparameter X wird durch den gegenwärtigen Wert von Z↑2−1, also durch die Zahl 3, ersetzt und sodann FNA(3) berechnet und gedruckt.

2. Statt

```
Y1=X1↑3+A*X1↑2+B*X1+C
Y2=X2↑3+A*X2↑2+B*X2+C
Y3=X3↑3+A*X3↑2+B*X3+C
PRINT  Y1, Y2, Y3
```

kann man kürzer schreiben:

```
DEF  FNA(X)=X↑3+A*X↑2+B*X+C
PRINT  FNA(X1), FNA(X2), FNA(X3)
```

3. Von einem Dreieck sind die Seite a und die Winkel α und β gegeben. Es ist ein Programm für die Berechnung der Seiten b und c und des Winkels γ zu erstellen.

Es gilt: $\frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma}$

Programm:

```
100 READ A,W1,W2
110 W3 = 180-(W1+W2)
120 DEF FNS(W) = SIN(W*1.74532925E-2)
130 S = A/FNS(W1)
140 B = S*FNS(W2)
150 C = S*FNS(W3)
160 PRINT "SEITEN:",A,B,C
170 PRINT "WINKEL:",W1,W2,W3
180 PRINT
190 GOTO 100
200 DATA 10,60,60, 15,10.4,12.8
```

Ausgabe:

```
SEITEN:          10              10.             10.
WINKEL:          60              60              60

SEITEN:          15              18.4093         32.7341
WINKEL:          10.4            12.8            156.8
```

ÜBUNGEN

Programmiere mit Hilfe der DEF-Anweisung:

5.15 $\frac{e^x - e^{-x}}{2}$ ..... [1]

5.16 $\frac{e^x + e^{-x}}{2}$ ..... [2]

5.17 $\frac{1}{\tan x}$ ..... [3]

5.18 $\sin\left(x + \frac{\pi}{100}\right)$

5.19 $\ln\left(x + \sqrt{x^2+1}\right)$ ..... [4]

5.20 $\ln\left(x + \sqrt{x^2-1}\right)$ ..... [5]

5.21 $\frac{1}{2}\ln\frac{1+x}{1-x}$ ..... [6]

---

[1] $\sinh x = \frac{e^x - e^{-x}}{2}$

[2] $\cosh x = \frac{e^x + e^{-x}}{2}$

[3] $\cot x = \frac{1}{\tan x}$

[4] $\operatorname{arsinh} x = \ln\left(x + \sqrt{x^2+1}\right)$

[5] $\operatorname{arcosh} x = \ln\left(x + \sqrt{x^2-1}\right)$

[6] $\operatorname{artanh} x = \frac{1}{2}\ln\frac{1+x}{1-x}$

6. EINGABE UND AUSGABE

Lernziele

Den Aufbau der Leseanweisung und die Form einer Datenzeile angeben können.

Angeben können, wann es zu einem Programmabbruch bei der Ausführung einer READ-Anweisung kommt.

Den Aufbau und die Wirkung der INPUT-Anweisung erklären können.

Den Aufbau der PRINT-Anweisung angeben können.

Die Wirkung der PRINT-Anweisung erklären können.

Die Bedingungen für

Schreibkopfrücklauf und Zeilenvorschub,
Schreiben in der nächsten Spalte,
Schreiben in der nächsten Position

angeben können.

Das Lesen und Drucken von alphamerischen Konstanten erklären können.

Den Zweck der formatisierten Ausgabe angeben können.

Die Form von Konstantenschablonen, Ganzzahlschablonen, Fixkommaschablonen und Gleitkommaschablonen angeben können.

Den Aufbau und die Wirkung der PRINTUSING-Anweisung erklären können.

6.1. READ-ANWEISUNG UND DATA-ANWEISUNG

Die Anweisung

```
DATA  Zahl 1, Zahl 2, Zahl 3, . . .
```

bewirkt, daß die Konstanten (hier Zahl 1, Zahl 2, Zahl 3, . . . genannt) in einem Datenblock gespeichert werden.

Einfachheitshalber schreiben wir DATA-Anweisungen als letzte Anweisungen im Programm.

DATA-Anweisungen können jedoch an beliebiger Stelle im Programm stehen, auch vor den READ-Anweisungen.

Die Anweisung

```
READ  Variable 1, Variable 2, . . .
```

bewirkt, daß Variable 1 mit dem nächsten Wert des Datenblocks, Variable 2 mit dem folgenden Wert usw. belegt wird.

Wenn der Datenblock nicht mehr genug Konstanten für die Belegung der Variablen enthält, wird die Rechnung abgebrochen.

Am Terminal wird die Abbruchmeldung

OUT OF DATA IN *n*

bekanntgegeben.

6.2. INPUT-ANWEISUNG

Mit Hilfe der Anweisung

INPUT Variable, 1, Variable 2, ...

kann man während des Programmablaufs Daten in den Rechner eingeben[1].

BEISPIEL

Programm:

```
100 PRINT "EINGABE VON R"
110 INPUT R
120 PRINT "INHALT =";3.14159*R↑2
```

Ausgabe:

```
EINGABE VON R
? 10
INHALT = 314.159
```

6.3. PRINT-ANWEISUNG

Allgemeine Form der PRINT-Anweisung:

PRINT Anweisungsliste	oder	PRINT

Die Elemente der Anweisungsliste können durch Beistrich oder Strichpunkt voneinander getrennte Zahlen und/oder Variablen und/oder Terme und/oder Texte sein.

Jedes Element der Anweisungsliste ist ein Druckauftrag!

Die PRINT-Anweisung bewirkt:

Schreibkopfrücklauf und Zeilenvorschub,

Zeileneinteilung,

Ausdrucken von Texten,

Ausdrucken des augenblicklichen Werts von Variablen,

Berechnung und Ausdrucken der Werte der in der Anweisungsliste stehenden Terme.

[1] INPUT-Anweisungen sind für Lehrprogramme und für Demonstrationsprogramme zu empfehlen. In diesem Buch werden sie nicht verwendet, weil sie kostspieliger als READ- und DATA-Anweisungen sind.

PRINT bewirkt 1 Zeile Vorschub, wenn der Schreibkopf in der Position 0 steht,

bewirkt den Beginn einer neuen Zeile, wenn sich der Schreibkopf in einer anderen Position befindet.

PRINT T bewirkt Berechnen und Drucken des Werts von T, Wagenrücklauf und Zeilenvorschub.

Die nächste PRINT-Anweisung wird daher in der nächsten Zeile ab Position 0 ausgeführt[1].

PRINT T, bewirkt Berechnen und Drucken des Werts von T.

Die nächste PRINT-Anweisung wird in der nächsten Spalte ausgeführt.

Spalte 2 ab Position 15, Spalte 3 ab Position 30,
Spalte 4 ab Position 45, Spalte 5 ab Position 60,
Spalte 1 der nächsten Zeile ab Position 0 usw.

Analog wirkt ein Beistrich als Trennzeichen zweier Elemente der Anweisungsliste[2].

PRINT T; bewirkt Berechnen und Drucken des Werts von T.

Der Strichpunkt bewirkt, daß der Schreibkopf nach dem Drucken eine, zwei oder drei Leerstellen erzeugt.

Analog wirkt ein Strichpunkt als Trennzeichen zweier Elemente der Anweisungsliste.

PRINT "X="; bewirkt das Drucken des Textes X=

Der Strichpunkt bewirkt, daß der Schreibkopf in der nächsten Position stehenbleibt.

BEISPIELE

1. PRINT X, Y, "KONTROLLWERT Z= ";Z bewirkt:

   Drucken des Werts von X ab Position 0,
   Drucken des Werts von Y ab Position 15,
   Drucken des Textes KONTROLLWERT Z= ab Pos. 30.

   Nach dem Gleichheitszeichen ist 1 Leerstelle einprogrammiert, dann wird der Wert von Z gedruckt.

2. Das Programm

```
100 READ A$
110 PRINT "MONAT:";A$
120 DATA"DEZEMBER 1974"
```

   bewirkt die Ausgabe

```
MONAT:DEZEMBER 1974
```

---

[1] Bei unserm Terminal hat jede Druckzeile 72 Positionen, die wir mit 0, 1, 2, ..., 71 numerieren.

[2] PRINT A, B, C bewirkt das Drucken von A ab Position 0, das Drucken von B ab Position 15 und das Drucken von C ab Position 30.

6.4. FORMATISIERTE AUSGABE[1]

Die durch die PRINT-Anweisungen bewirkten Druckanordnungen wurden in 6.3. angegeben.

Wenn eine andere Druckanordnung gewünscht wird, so verwendet man das Anweisungspaar

PRINTUSING und :

Allgemeine Form einer PRINTUSING-Anweisung:

PRINTUSING Zeilennummer , Anweisungsliste

oder

PRINTUSING Zeilennummer

Allgemeine Form einer Doppelpunkt-Anweisung:

Zeilennummer: Schablone

Beachte: Zwischen Zeilennummer und Doppelpunkt darf keine Leerstelle sein!

Die Schablone darf bis zu 72 Zeichen enthalten.

Wir unterscheiden:

Konstante Schablonen

z. B.: `150: X "KURZ(X)`

Ganzzahlschablonen

z. B.: `1:##  ####  ####`

Beachte:

Das Datenfeld einer Ganzzahl besteht nur aus #-Zeichen.

Die Zahlen werden rechtsbündig geschrieben.

Nichtganze Zahlen werden abgeschnitten, also nicht gerundet.

Wenn die Zahl mehr Stellen hat, als für sie bereitgestellt wurden, so wird das Feld (und damit alle rechts von diesem Feld befindlichen Felder) nach rechts erweitert.

Für das Vorzeichen muß eine Stelle vorgesehen werden!

Wenn $|z| > 1073741823$ ist, so wird für z ein Sternchen gedruckt.

---

[1] *Hinweis:* Hier werden nur die Grundzüge der formatisierten Ausgabe behandelt. Wie man in der Praxis die formatisierte Ausgabe anwendet, wird im Folgeband BASIC-Praxis gezeigt.

Fixkommaschablonen

z. B.: 2: ##.## #.#### #.#### #####.##

Beachte:

Das Datenfeld besteht aus Nummernzeichen, die einen Dezimalpunkt einschließen.

Die Zahlen werden rechtsbündig geschrieben.

Wenn zu wenig Stellen angegeben sind, so wird die Zahl gerundet ausgegeben.

Wenn die Zahl mehr Stellen hat, als für sie vorgesehen wurden, so wird das Feld nach rechts erweitert.

Für das Vorzeichen muß eine Stelle vorgesehen werden!

Gleitkommaschablonen

z. B.: 3:##.####↑↑↑↑ ##.##↑↑↑↑

Beachte:

Für den Dezimalteil stehen #-Zeichen.

Für den Exponentialteil steht ↑↑↑↑.

Der Dezimalteil wird gerundet.

Für das Vorzeichen des Dezimalteils muß eine Stelle vorgesehen werden.

BEISPIEL

Programm:

```
1:####  ####  ####  ####
2:X = ###.##  Y = ##.##  Z = ##.####↑↑↑↑
100 PRINTUSING 1, -456, 12.9, -12.9, 2.13E2
110 PRINTUSING 1, 2E9, 8, -1125, 4
120 PRINTUSING 2, -14.567, 6, .001234567
130 PRINTUSING 2, -2012.1, 3.14, 1789.63
```

Ausgabe:

```
-456    12   -12   213
   *     8  -1125     4
X = -14.57  Y =  6.00  Z =  1.235E-03
X = -2012.10  Y =  3.14  Z =  1.790E+03
```

7. PROGRAMMABLAUFPLAN

Lernziele

Die Symbole für Beginn und Ende, Eingabe, Ausgabe, Operation und Konnektor zeichnen und erklären können.

Zeichenkonventionen für die Verbindung der Symbole der Flußprogrammtechnik angeben können.

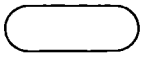
Wiederholtes Eingeben und Ausdrucken von Daten programmieren und graphisch darstellen können.

Ein Programm mit wiederholten Berechnungen schreiben und graphisch darstellen können.

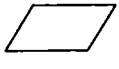
Bei komplizierten Aufgaben ist es ratsam, vor der Programmierung den Programmablauf bildhaft zu skizzieren. Die graphische Darstellung dieser Programmschritte nennt man *Programmablaufplan, Flußdiagramm, flow chart* oder *flow diagram.*

Anhand eines Programmablaufplanes läßt sich der zeitliche Ablauf des Programms leicht verfolgen und beurteilen. Logische Fehler können auf diese Weise schnell gefunden werden.

Wir werden von dieser Darstellungsart später Gebrauch machen. Dabei werden wir folgende Symbole verwenden:

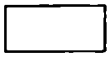


Grenzstelle
z. B.: Beginn, Ende

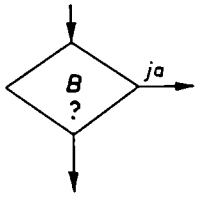


Eingabe

Ausgabe am Blattschreiber

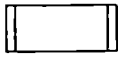


Allgemeine Operation
z. B.: Rechnen, Vertauschen



Verzweigung
Der weitere Programmablauf hängt von einer Bedingung, hier mit B bezeichnet, ab. Wenn diese erfüllt ist, so schlägt der Rechner den JA-Weg ein, andernfalls den NEIN-Weg.

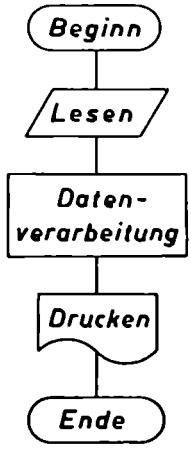
Übergangsstelle (Konnektor)
Wir verwenden meistens die Zeilennummern zur Kennzeichnung der entsprechenden Übergangsstelle.



Unterprogramm (Näheres siehe Abschnitt 12.)

Um eine möglichst einfache und übersichtliche Darstellung zu erreichen, legen wir fest: Wenn nicht ausdrücklich durch Pfeile etwas anderes festgelegt wird, so sind die Verbindungslinien von oben nach unten orientiert.

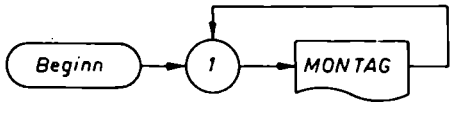
Wir wollen nun einige einfache Programmablaufpläne besprechen.



Programme ohne Verzweigungen heißen *gestreckte Programme*. Bei ihnen wird jede Anweisung nur einmal durchlaufen. Ihr prinzipieller Aufbau ist aus dem nebenstehenden Programmablaufplan zu ersehen.

Programmteile, die mehrmals durchlaufen werden, heißen *Schleifen*.

Der Programmablaufplan



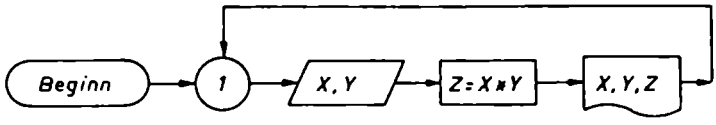
ist eine Veranschaulichung des Programms

```
100 PRINT "MONTAG"
110 GOTO 100
```

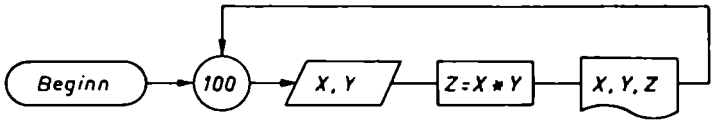
das das ununterbrochene Drucken des Textes MONTAG bis zum Abbruch durch den Operator verursacht.

Die Programmablaufpläne

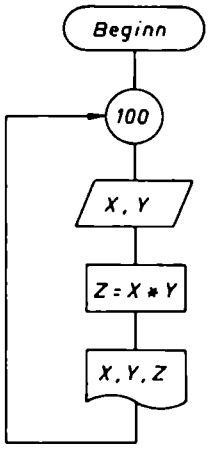
a)



b)



c)



sind jeweils
eine Veranschaulichung
des Programms:

```
100 READ X,Y
110 Z = X*Y
120 PRINT X,Y,Z
130 GOTO 100
140 DATA 2,4, 5,5, 7
```

Ausgabe:

```
 2                4                8
 5                5                25
OUT OF DATA IN 100
```

Beachte: Das Programm wird durch *Datenerschöpfung* beendet.

8. VERZWEIGUNGEN UND SCHLEIFEN

Lernziele

Den Aufbau und die Wirkung der IF-Anweisung erklären können.

Die in einer IF-Anweisung zulässigen Vergleichssymbole angeben können.

Mit IF-Anweisungen Schleifen und den Aussprung aus ihnen programmieren können.

Eine Programmverzweigung ohne Wiedervereinigung programmieren können.

Den Aufbau der FOR-Anweisung und der NEXT-Anweisung angeben können.

Die Ausführung von FOR-Schleifen erklären können.

Das Überspringen von FOR-Schleifen erklären können.

Die Ausführung von FOR-Schleifen nachvollziehen können.

Logische Fehler in FOR-Anweisungen erklären können.

Den Aufbau und die Wirkung von geschachtelten FOR-Schleifen erklären können.

Die Symbole für IF-Anweisungen und für FOR-Schleifen angeben können.

Programmablaufpläne mit Verzweigungen und Schleifen zeichnen können.

8.1. IF-ANWEISUNG

Mit Hilfe der *bedingten Sprunganweisung*[1]

```
IF  T1<T2  THEN  n
```

kann man den Ablauf eines Programms so steuern, wie es die Logik des Programms erfordert. Wenn der Termwert von T_1 kleiner ist als der Termwert von T_2, so setzt das Programm mit der Zeile n fort. Wenn diese Bedingung nicht erfüllt ist, so wird mit der unmittelbar auf

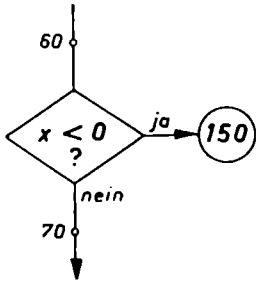
`IF T1<T2 THEN n` folgenden Anweisung fortgesetzt[2].

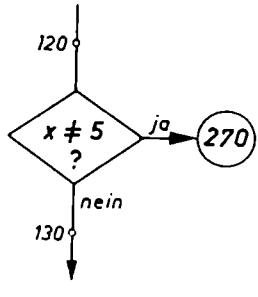
[1] Man spricht dann von einem bedingten Sprung, wenn der Sprung von einer Bedingung abhängig ist. Zum Beispiel: Springe nach der Zeile 170 genau dann, wenn $x < 1{,}5$ ist!

[2] if ≙ wenn; then ≙ dann

Als Relationszeichen in IF-Anweisungen sind zugelassen	Entsprechendes mathematisches Zeichen	Beispiel
=	=	A=3*B
<	<	X<1E−5
<=	≦	A<=2*SIN(X)
>	>	(1−4.5*X↑(1/3))>TAN(X)
>=	≧	Y>=Ø
<>	≠	3.2*X−2.8<>Ø

Im *Programmablaufplan* werden bedingte Verzweigungen durch auf eine Spitze gestellte Rhomben dargestellt.






```
60  IF X<Ø THEN 15Ø
7Ø
```

Wenn $x<0$ ist, so wird der JA-Weg eingeschlagen. Der Rechner setzt mit der Anweisung 15Ø fort.

Wenn $x \geqq 0$ ist, so wird der NEIN-Weg eingeschlagen; der Rechner setzt mit der nächsten Anweisung, also mit 7Ø ..., fort.

```
12Ø  IF X<>5 THEN 27Ø
13Ø  .....
```

Wenn $x \neq 5$ ist, so wird der JA-Weg eingeschlagen. Der Rechner setzt mit der Anweisung 27Ø fort.

Wenn $x=5$ ist, so wird der NEIN-Weg eingeschlagen; der Rechner setzt mit der Anweisung 13Ø ... fort.

Beim Testen von Programmen mit Verzweigungen muß jeder Programmweg durchlaufen werden.
Um die Auswirkung von Rundungsfehlern bei der IF-Anweisung zu vermeiden, setzt man fest:

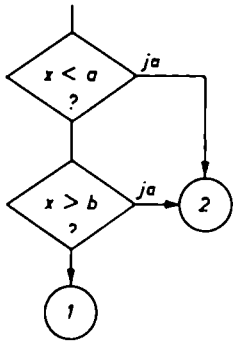
$x=y$ ist schon dann erfüllt, wenn $|x-y| < |x| \cdot 10^{-8}$ ist.

BEISPIELE

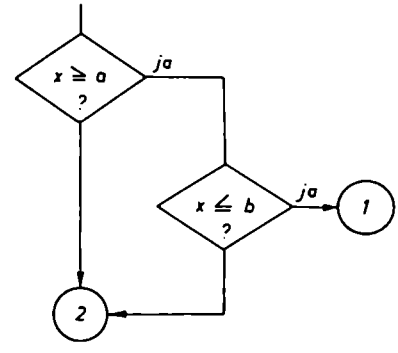
1. Es ist ein Programmablaufplan für folgendes Teilproblem zu zeichnen: Es soll überprüft werden, ob x im Intervall $[a, b]$ liegt.

Lösung:

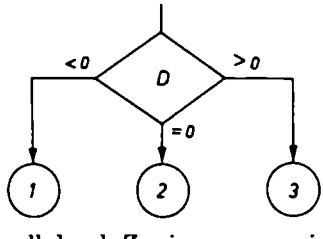
Marke 1: x liegt in $[a, b]$



Marke 2: x liegt nicht in $[a, b]$

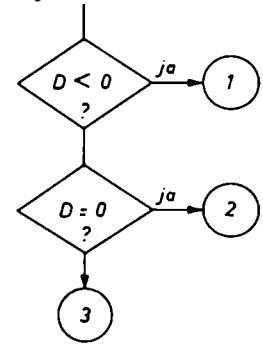


2. Die *Dreiwegeverzweigung*



soll durch Zweiwegeverzweigungen ersetzt werden.

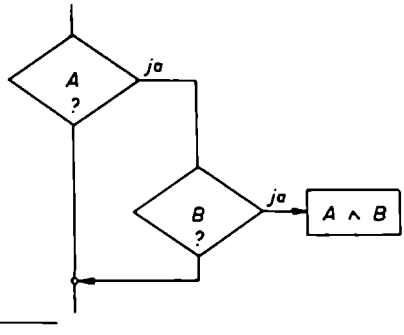
Lösung:



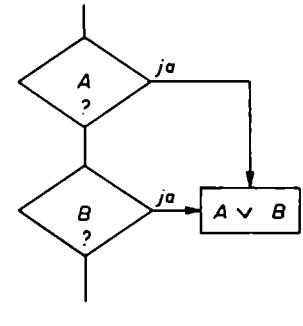
3. A und B seien Bedingungen.

Programmablaufplan für die

UND-Verknüpfung $A \wedge B$

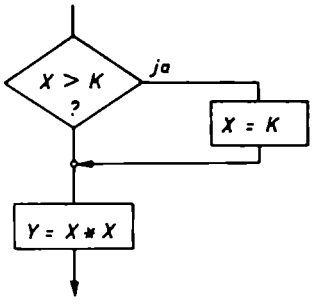


ODER-Verknüpfung $A \vee B$



[1] $A \wedge B$ Sowohl die Bedingung A als auch die Bedingung B ist erfüllt.
$A \vee B$ Entweder A oder B oder beide Bedingungen sind erfüllt.

4.



Unvorteilhafte Programmierung:

```
100 IF X > K THEN 120
110 GOTO 130
120 X = K
130 Y = X↑2
```

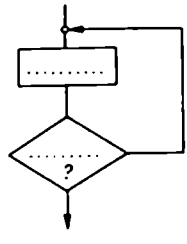
Besser:

```
100 IF X <= K THEN 120
110 X = K
120 Y = X↑2
```

Beachte: Der NEIN-Weg enthält hier keine eigenen Operationen, er dient nur zum Überspringen der Operation(en) des JA-Wegs.

In solchen Fällen formuliert man die IF-Anweisung so, daß die erfüllte Bedingung zum Überspringen führt.

Wenn im Anschluß an eine Verzweigung zu einer früheren Stelle im Programm gesprungen wird, so entsteht eine *Schleife.*

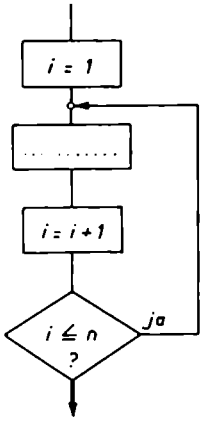


Wir unterscheiden:

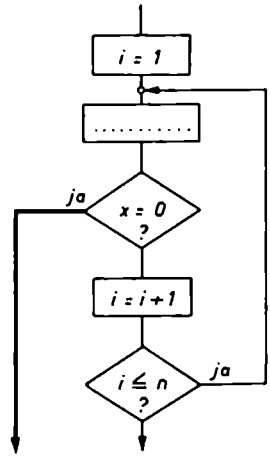
a) *Induktive Schleifen*

Ein Zähler kontrolliert die vorgegebene Anzahl der Schleifendurchläufe.

Induktive Schleife
mit Normalausgang:



Induktive Schleife
mit bedingtem Ausgang:



In beiden Fällen wird die Schleife nach n Durchläufen durch den „Normalausgang" verlassen.

Im zweiten Fall wird die Schleife, falls $x = 0$ ist, frühzeitig durch den „Notausgang" verlassen.

b) *Sukzessive Schleifen*

Die Schleife wird solange durchlaufen, bis eine *Datenendbedingung*[1] erfüllt ist.

c) *Iterative Schleifen*

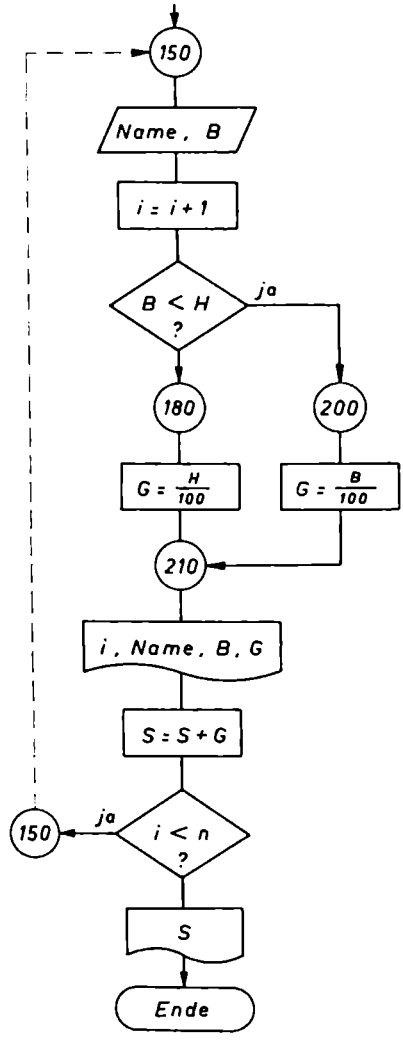
Sukzessive Schleifen, bei denen die Datenendbedingung eine Konvergenzbedingung ist, bezeichnet man als iterative Schleifen.

WEITERE BEISPIELE

5. Bis zu einem Monatsbruttogehalt von H ö. S beträgt der Gewerkschaftsbeitrag für öffentliche Angestellte 1% des Bezugs. Ab H ö. S sind monatlich $\frac{H}{100}$ ö. S zu entrichten[2].

Es sind der nebenstehende Programmablaufplan und ein Programm für die Erstellung einer Liste, die eine Nummer i, den Namen, das Bruttogehalt B und den Gewerkschaftsbeitrag G enthält, zu erstellen. Außerdem soll der von der n Mitglieder umfassenden Dienststelle abzuliefernde Gewerkschaftsbeitrag eines Monats ausgedruckt werden.

Hiebei hat die Beitragsabrechnung nach steigenden Nummern zu erfolgen, was bereits bei der Dateneingabe zu berücksichtigen ist.



[1] Zum Beispiel: X > 5, Ende einer Datei, ...

[2] März 1975: $H = 6100$

Programm:

```
100 PRINT "GEWERKSCHAFTSBEITRAG FUER ";
110 READ M$,H,N
120 PRINT M$
130 PRINT
140 PRINT "LFD.NR.","NAME","BRUTTO","BEITRAG"
150 READ A$,B
160 I = I+1
170 IF B < H THEN 200
180 G = H/100
190 GOTO 210
200 G = B*0.01
210 PRINT I,A$,B,G
220 S = S+G
230 IF I < N THEN 150
240 PRINT
250 PRINT "BEITRAGSSUMME=";S
260 DATA "JAENNER 1975", 6400, 8
270 DATA "WEISS KARL", 3850, "SCHWARZ FRANZ", 13420
280 DATA "NOWOTNY ALFONS",9830, "BITTER HANS",6140
290 DATA "GASSNER KARL",7580, "KURZ ELSA",15800
300 DATA "SMEJKAL WERNER ",5920,"AUBOECK FRANZ ",6200
```

Ausgabe:

```
GEWERKSCHAFTSBEITRAG FUER JAENNER 1975

LFD.NR.         NAME            BRUTTO          BEITRAG
 1              WEISS KARL       3850            38.5
 2              SCHWARZ FRANZ    13420           64
 3              NOWOTNY ALFONS   9830            64
 4              BITTER HANS      6140            61.4
 5              GASSNER KARL     7580            64
 6              KURZ ELSA        15800           64
 7              SMEJKAL WERNER   5920            59.2
 8              AUBOECK FRANZ    6200            62.

BEITRAGSSUMME= 477.1
```

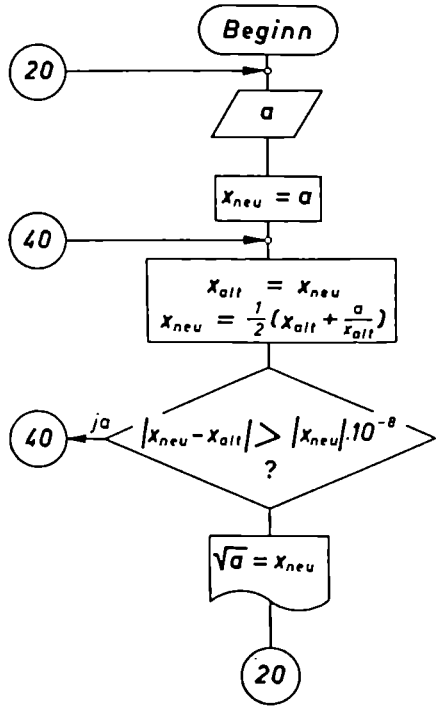
6. Es ist ein Programm für die Berechnung von $\sqrt{a}$ mit Hilfe der von Archimedes stammenden Näherungsformel

$$x_{neu} = \frac{1}{2}\left(x_{alt} + \frac{a}{x_{alt}}\right)$$

zu schreiben. Es soll a als erster Näherungswert x_0 verwendet werden[1] und die Iteration[2] so lange durchgeführt werden, bis zwei aufeinanderfolgende Näherungswerte auf 7 Stellen übereinstimmen, d. h.

$|x_{neu} - x_{alt}| \leqq |x_{neu}| \cdot 10^{-8}$

ist[3].



Programm:

```
20 READ A
30 X = A          ...4
40 Y = X
50 X = (Y+A/Y)/2
60 IF ABS((X-Y)/X) > 1E-8 THEN 40
70 PRINT "A =";A, "WURZ(A) =";X
80 GOTO 20
90 DATA 10000, .81, .081
```

Ausgabe:

```
A =  10000        WURZ(A) =  100
A =  .81          WURZ(A) =  .9
A =  .081         WURZ(A) =  .284605

OUT OF DATA  IN 20
```

[1] $a \neq 0$ wird vorausgesetzt!

[2] Unter *Iteration* versteht man die wiederholte Anwendung desselben Rechenverfahrens zur Gewinnung einer Näherungslösung.

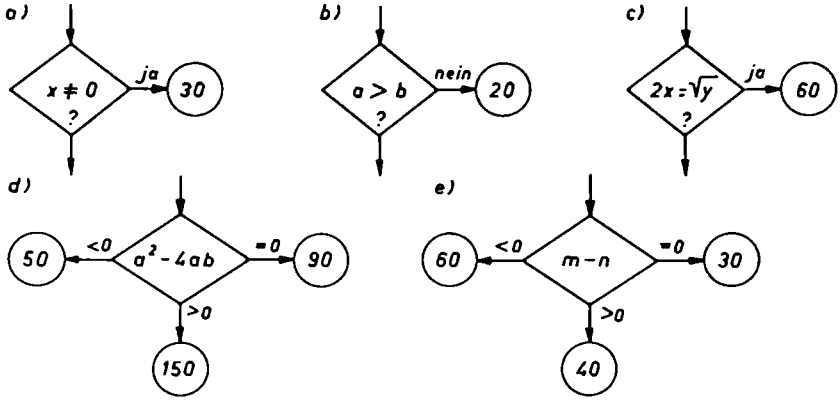
[3] 1. Schritt: $x = \frac{1}{2}(a+1)$ 2. Schritt: $x = \frac{1}{2}\left[\frac{a+1}{2} + \frac{2a}{a+1}\right]$

Für $a = 3$ erhält man so: $x = 2$, $x = \frac{1}{2}\left(2 + \frac{3}{2}\right) = 1{,}750$ usw.

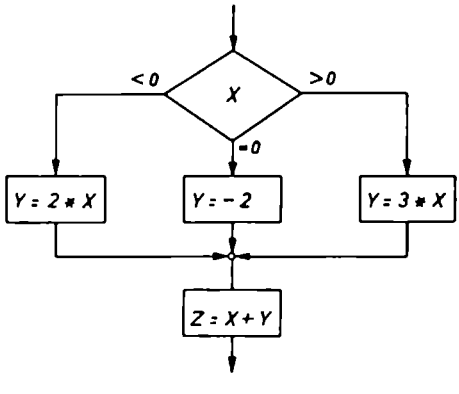
[4] $X = x_{neu}$; $Y = x_{alt}$

ÜBUNGEN

8.01 Wie lauten die Anweisungen für die folgenden Zweiwegeverzweigungen bzw. Dreiwegeverzweigungen?



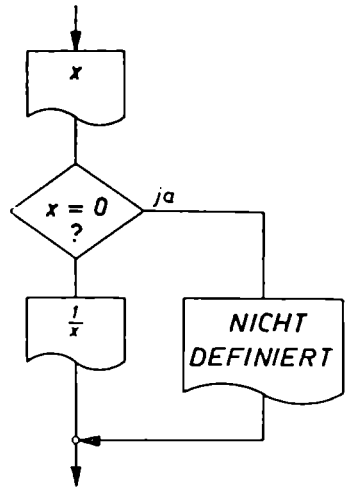
8.02 Wie lautet der zugehörige Programmausschnitt?



8.03 Es sind ein Programmablaufplan und ein BASIC-Programm für die Lösung der linearen Gleichung $ax+b=0$ zu erstellen.

8.04 Es sind ein Programm und ein Programmablaufplan für die Berechnung der Summe und des Produkts der ganzen Zahlen von a bis b zu erstellen. Es gelte: $a, b \in \mathbb{Z}$ und $a<b$.

8.05 Es ist ein Programm für das Ausdrucken einer Liste von $\frac{1}{x}$ für $x=-0{,}6$ bis $x=+0{,}6$ zu schreiben. Die Schrittweite sei $\Delta x=0{,}2$. Bei der Erstellung des Programms ist zu berücksichtigen, daß $\frac{1}{x}$ für $x=0$ nicht definiert ist.



8.06 Es ist ein Programm für die Berechnung von e^x mit Hilfe einer Iterationsschleife zu schreiben.

Anleitung:

Die Funktion $y=e^x$ kann durch die Taylor-Reihe

$$e^x=1+\frac{x}{1!}+\frac{x^2}{2!}+\ldots+\frac{x^n}{n!}+\ldots$$

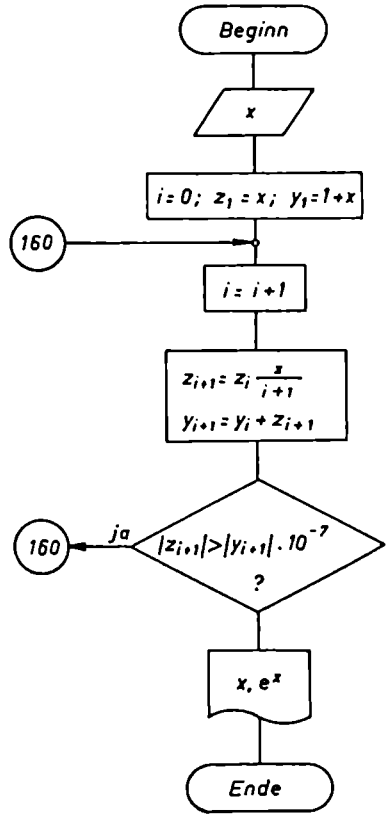
dargestellt werden. Diese Reihe konvergiert für alle endlichen x-Werte.

In einem bestimmten Intervall, z. B. in $0\leqq x<1$, kann e^x durch die ersten $n+1$ Glieder der Reihe, also durch ein Polynom n-ten Grades, dargestellt werden.

Leicht sind folgende Gesetzmäßigkeiten zu erkennen:

$$y = 1+z_1+z_2+\ldots+z_n$$
$$z_{i+1} = z_i\cdot\frac{x}{i+1}$$
$$y_i = 1+z_1+\ldots+z_i$$
$$y_{i+1} = y_i+z_{i+1}$$

Die Rechnung soll als beendet gelten, wenn $|z_{i+1}|<|y|\cdot 10^{-7}$ ist.



8.2. FOR-ANWEISUNG UND NEXT-ANWEISUNG

Durch das Anweisungspaar FOR und NEXT ist die Programmierung von Schleifen möglich.

FOR ist der Anfang der Schleife.

NEXT ist das Ende der Schleife.

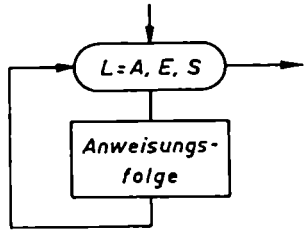
Allgemeine Form einer FOR-Schleife:

```
FOR L = A TO E STEP S

Anweisungsfolge

NEXT L
```

Programmablaufplan:



L heißt *Laufvariable*,
A *Anfangswert*,
E *Endwert*,
S *Schrittweite (Inkrement)* der Schleife.

In BASIC muß L eine Variable (AØ, ..., Z9) sein, A, E, S dürfen Terme sein!

Beim 1. Durchlauf der Schleife wird L mit dem Wert von A belegt, beim 2. Durchlauf mit dem Wert A+S usw.

Wenn der Wert der Laufvariablen L größer wird als der Endwert, dann wird die der Schleife im Programm folgende Anweisung durchgeführt.

FOR L=A TO E ist gleichwertig FOR L=A TO E STEP 1

BEISPIELE

1. Das Programm

```
10 FOR X = 1 TO 1.05 STEP .01
20 PRINT X,X↑(1/3)
30 NEXT X
```

bewirkt die Ausgabe:

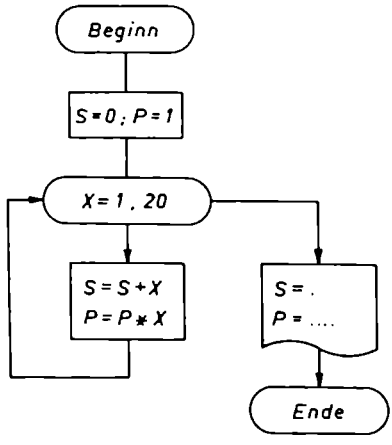
```
 1                    1
 1.01                 1.00332
 1.02                 1.00662
 1.03                 1.0099
 1.04                 1.01316
 1.05                 1.0164
```

Die Anweisung 10 bewirkt, daß die Laufvariable X mit dem Anfangswert 1 belegt wird. Mit diesem Wert wird die Anweisung 20 durchgeführt.

Die NEXT-Anweisung in Zeile 30 bewirkt, daß X so lange um die Schrittweite 0.01 erhöht wird, bis der Endwert 1.05 erreicht wird.

2. Es ist ein Programm zu schreiben für die Berechnung der Summe und des Produkts der ganzen Zahlen von 1 bis 20.

Programmablaufplan:



Programm:

```
100 P = 1          ...[1]
110 FOR X = 1 TO 20
120 S = S+X
130 P = P*X
140 NEXT X
150 PRINT "SUMME =";S,"PRODUKT =";P
```

Ausgabe:

```
SUMME = 210      PRODUKT = 2.43290E+18
```

[1] In den Abschnitten 2.4. und 4. wurde darauf hingewiesen, daß jede numerische Variable am Beginn der Rechnung mit dem Wert 0 belegt wird.
Würde die Variable P vor Beginn der Schleife nicht mit der Zahl 1 belegt werden, so wäre der Wert von P von Anfang bis Ende der Schleife gleich Null.

3. Es ist ein Programm für die Berechnung der statistischen Maßzahlen von n Daten (bei k Klassen) zu erstellen.

Lösung:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} = D + \frac{\sum_{j=1}^{k} f_j \Delta x_j}{\sum_{j=1}^{k} f_j} \qquad D \ldots \text{gewählter, vorläufiger Mittelwert}$$

$$s^2 = \frac{1}{n-1}\left[\sum_{j=1}^{k} f_j (\Delta x_j)^2 - \frac{1}{n}\left(\sum_{j=1}^{k} f_j \Delta x_j\right)^2\right] \qquad \Delta x_j = x_j - D$$

Programm:

```
100 READ K, D
110 FOR J = 1 TO K
120 READ X, F
130 N = N+F
140 X0 = X-D
150 G = F*X0
160 S = S+G
170 Q = Q+G*X0
180 NEXT J
190 V = (Q-S*S/N)/(N-1)
200 PRINT "N  = "; N
210 PRINT "XM = "; D+S/N, "V = "; V, "S = "; SQR(V)
220 DATA 8, 830
230 DATA 827,5
240 DATA 828,7
250 DATA 829,24
260 DATA 830,9
270 DATA 831,2
280 DATA 832,4
290 DATA 833,0
300 DATA 834,1
```

Ausgabe:

```
N  =  52
XM =  829.25    V =  1.9951     S =  1.41248
```

4. Es ist ein Programm für die Berechnung des Skalarprodukts zweier Vektoren zu erstellen.

Der Vektor x habe die Koordinaten $x_1, x_2, \ldots, x_n$.
Der Vektor y habe die Koordinaten $y_1, y_2, \ldots, y_n$.

Lösung:

Für das Skalarprodukt S gilt definitionsgemäß:

$$S = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + x_2 y_2 + \ldots + x_n y_n$$

Programm:

```
100 READ N
110 FOR I = 1 TO N
120 READ X, Y
130 S = S+X*Y
140 NEXT I
150 PRINT "SKALARPRODUKT =";S
160 DATA 5, 3, 2, 5, 1, 0, 3, 4, 2, 6, 7
```

Bemerkung zu 160: Es ist $n=5$, $x_1=3$, $y_1=2$, $x_2=5$, $y_2=1$, ...

Ausgabe:

```
SKALARPRODUKT =  61
```

Für die Programmierpraxis ist wichtig:

Der zuletzt verwendete Wert von L bleibt beim Verlassen der Schleife erhalten.

Die Schleifenvariable L darf im Schleifeninnern geändert werden.

Schleifen mit Laufanweisungen der Art

```
100 FOR Y=5 TO 10 STEP -1
```

```
150 FOR X=2 TO -4
```

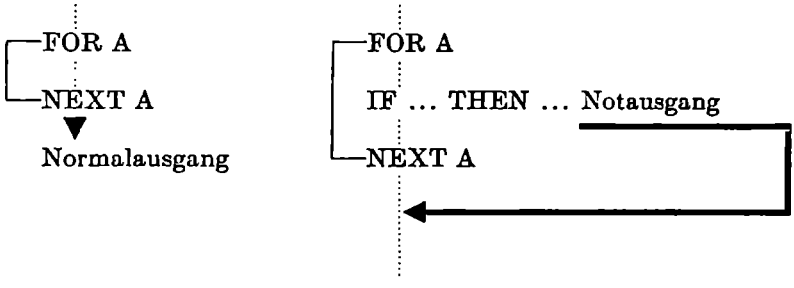
```
180 FOR K=10 TO 2 STEP 3
```

werden vom Rechner übersprungen.

Schrittweiten, die der Rechner als Ø deutet (bei Mark I/7 sind dies alle Zahlen z, für die $|z| < 4.318085E-78$ gilt), führen, falls der Wert der Laufvariablen nicht im Schleifeninnern geändert wird, zu *endlosen Schleifen.*

Üblicherweise wird eine Schleife nach vollständiger Verarbeitung über den „Normalausgang" verlassen.

Es ist jedoch möglich, aus der Schleife durch einen „Notausgang" herauszuspringen, z. B. durch eine bedingte Sprunganweisung in der Schleife. Es ist zu beachten, daß in diesem Fall die Laufvariable ihren zuletzt angenommenen Wert beibehält.

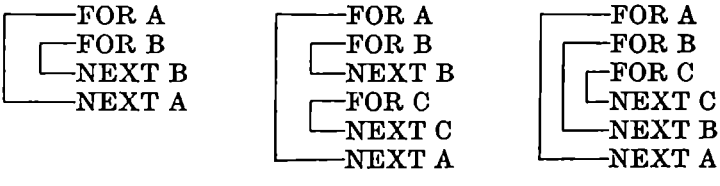


Nach einem Sprung aus der Schleife ist der Rücksprung in die Schleife gestattet.

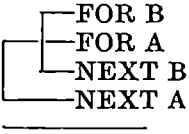
Von einer außerhalb einer Schleife gelegenen Zeile darf sonst nicht in die Schleife (auch nicht zur NEXT-Anweisung) gesprungen werden.

Wenn mehrere Schleifen ineinandergeschachtelt sind, so spricht man von einem *Schleifennest*[1].

Erlaubt sind z. B. folgende Schleifennester:



Nicht erlaubt sind *einander schneidende Schleifen*, z. B.:



[1] Auch die Ausdrücke *eingebettete Schleifen* und *ineinandergeschachtelte Schleifen* sind üblich.

WEITERE BEISPIELE

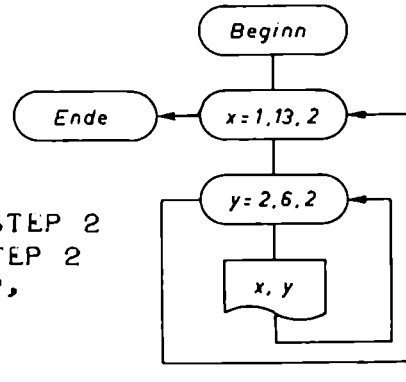
5. Unter der *Produktmenge* $X \times Y$, auch kartesisches Produkt genannt, versteht man die Menge aller geordneten Paare (x,y), deren erste Komponente x ein Element aus X und deren zweite Komponente y ein Element aus Y ist.

$$X \times Y = \{(x_1, y_1), (x_1, y_2), \ldots, (x_n, y_s)\}$$

Gegeben seien: $X = \{1, 3, 5, 7, 9, 11, 13\}$ und $Y = \{2, 4, 6\}$

Es ist ein Programm zu erstellen, das das Ausdrucken von $X \times Y$, also von $\{(1, 2), (1, 4), (1, 6), (3, 2), \ldots, (13, 6)\}$ bewirkt.

Programmablaufplan:



Programm:

```
10 FOR X = 1 TO 13 STEP 2
20 FOR Y = 2 TO 6 STEP 2
30 PRINT "(";X;Y;")",
40 NEXT Y
50 NEXT X
```

Ausgabe:

```
( 1  2 )    ( 1  4 )    ( 1  6 )    ( 3  2 )    ( 3  4 )
( 3  6 )    ( 5  2 )    ( 5  4 )    ( 5  6 )    ( 7  2 )
( 7  4 )    ( 7  6 )    ( 9  2 )    ( 9  4 )    ( 9  6 )
( 11  2 )   ( 11  4 )   ( 11  6 )   ( 13  2 )   ( 13  4 )
( 13  6 )
```

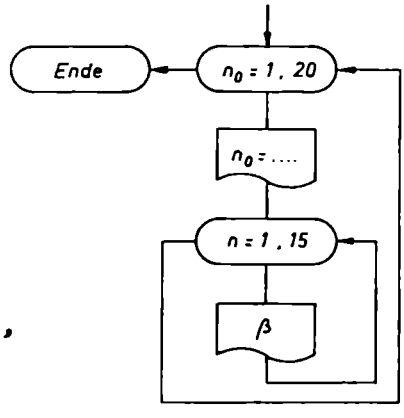
6. Für die Grenzwertermittlung gebrauchter Anlagen gilt:

$$W = W_0 \frac{1}{1 + \frac{n}{n_0}} = W_0 \cdot \beta$$

Bezeichnungen: W ... Wert nach n Jahren
β ... Bewertungsfaktor
n ... abgelaufene Nutzungsdauer
W_0 ... Anschaffungswert
n_0 ... angenommene Nutzungsdauer

Es ist ein Programm für die Berechnung von β für $n_0 = 1, 2, \ldots,$ 20 Jahre und $n = 1, 2, \ldots, 15$ Jahre zu erstellen.

Programm:




```
100 PRINT "BETA"
110 FOR N0 = 1 TO 20
120 PRINT
130 PRINT "N0 = ";N0
140 FOR N = 1 TO 15
150 PRINT 1/(1+N/N0),
160 NEXT N
170 NEXT N0
```

Ausgabe:

```
BETA

N0 =  1
 .5             .333333        .25            .2             .166667
 .142857        .125           .111111        .1             9.09091E-02
 8.33333E-02    7.69231E-02    7.14286E-02    6.66667E-02    .0625

N0 =  2
 .666667        .5             .4             .333333        .285714
```

ÜBUNGEN

8.07 Welche logischen Fehler enthalten die folgenden FOR-Anweisungen?

a) 100 FOR I=K TO K STEP 2
b) 200 FOR K=2 TO 8 STEP 1E−12
c) 300 FOR L=−3 TO −10

8.08 Es ist ein Programm für das Ausdrucken der Tabelle

X(GRAD)	X(BOGEN)	SIN(X)	SIN(X+30)
5	8.72665E-02	8.71557E-02	.573576
10	.174533	.173648	.642788
15	.261799	.258819	.707107
20	.349066	.34202	.766044

zu erstellen.

8.09 Es ist ein Programm für die Berechnung der Abzinsungsfaktoren v^n für $p=5\%$ und $n=1, 2, \ldots, 9, 10$ Jahre zu schreiben. Es gilt:

$$v=1:r=1:\left(1+\frac{p}{100}\right)$$

8.10 Die Übung 8.05 ist mit Hilfe einer FOR-Anweisung zu lösen.

8.11 Es ist ein Programm für das Drucken von 24 höchstens vierstelligen Zufallszahlen in Rechteckform (3 Zeilen und 8 Spalten) zu schreiben[1].

8.12 Es ist ein Programm für das Ausdrucken der folgenden Tabelle, die Werte der Binomialkoeffizienten $\binom{n}{k}$ enthält, zu schreiben[1].

N	K = 0	1	2	3	4	5	6	7	8	9
1	1	1								
2	1	2	1							
3	1	3	3	1						
4	1	4	6	4	1					
5	1	5	10	10	5	1				
6	1	6	15	20	15	6	1			
7	1	7	21	35	35	21	7	1		
8	1	8	28	56	70	56	28	8	1	
9	1	9	36	84	126	126	84	36	9	1

Anleitung: $\binom{n}{0}=1$; $\binom{n}{k}=\binom{n}{k-1}\cdot\frac{n-k+1}{k}$

[1] *Beachte:* Im Standardformat werden 5 Spalten erzeugt. Um die gewünschte Spaltenanzahl zu erhalten, muß die PRINTUSING-Anweisung verwendet werden.

9. VARIABLENFELDER

Lernziele

Die Begriffe *indizierte Variable*, *Index* und *Variablenfeld* erklären können.

Den Zweck von Variablenfeldern erklären können.

Den Aufbau einer indizierten Variablen erklären können.

Den Aufbau und die Wirkung einer DIM-Anweisung erklären können.

Programme mit indizierten Variablen schreiben können.

9.1. INDIZIERTE VARIABLEN ...[1]

Variablen der Form a_0, a_1, a_2, ..., a_n, ... bezeichnet man als *einfach indizierte Variablen.*

In BASIC schreibt man A(Ø) für a_0, A(1) für a_1, A(N) für a_n usw.

Variablen mit zwei Indizes bezeichnet man als *zweifach indizierte Variablen.*

Die zweifach indizierten Variablen a_{11}, a_{12}, ..., a_{mn}, c_{54}, r_{32} werden in BASIC durch A(1, 1), A(1, 2), ..., A(M, N), C(5, 4), R(3, 2) dargestellt.

Darstellung von indizierten Variablen in BASIC:	
einfach indizierte Variablen:	Buchstabe (**T**)
zweifach indizierte Variablen:	Buchstabe (**T**$_1$, **T**$_2$)

Beachte: Wird ein Index als nichtnegative Dezimalzahl eingegeben oder errechnet, so wird nur der ganzzahlige Teil berücksichtigt. Es wird somit A(7.82) zu A(7).

Die Verwendung desselben Buchstabens für eine einfach indizierte und für eine zweifach indizierte Variable im selben Programm ist verboten! Es darf jedoch derselbe Buchstabe für eine indizierte und außerdem für eine nichtindizierte Variable verwendet werden; z. B.: A und A(5) sind gestattet; A(5) und A(3, 2) sind nicht gestattet.

[1] Wir beschränken uns hier auf numerische Variablen.

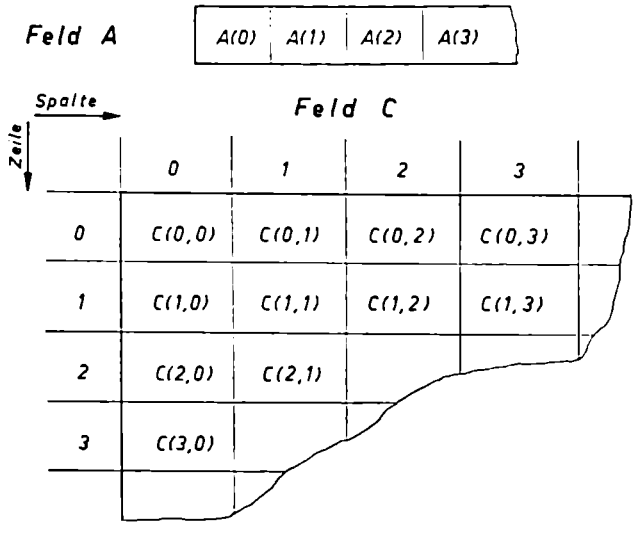
9.2. DIMENSIONIERUNG VON FELDERN

Man bezeichnet die Gesamtheit der Variablen A(Ø), A(1), A(2), ... als *Feld* A und die Gesamtheit der Variablen C(Ø, Ø), C(Ø, 1), ... als *Feld* C.

> Die Gesamtheit der indizierten Variablen, die sich nur in ihren Indizes unterscheiden, heißt *Variablenfeld* oder kurz *Feld*.

Beachte: Der zur Kennzeichnung der indizierten Variablen verwendete Buchstabe ist der *Name des Feldes*!

Der Index einer indizierten Variablen gibt ihre Stellung innerhalb des Feldes an!



Die Größe eines Feldes muß vor Verwendung einer ihrer indizierten Variablen vereinbart werden, wenn die Indexmaximalwerte größer als 10 sind.

Durch die Vereinbarung

```
100 DIM A(50), R(20), C(5, 4)
```

werden für das Feld A (50 + 1) Speicherplätze,
für das Feld R (20 + 1) Speicherplätze,
für das Feld C (5 + 1)*(4 + 1) Speicherplätze

reserviert, die unter dem Namen A(Ø), ..., A(5Ø), R(Ø), ..., R(2Ø) und C(Ø, Ø), ..., C(5, 4) aufrufbar sind.

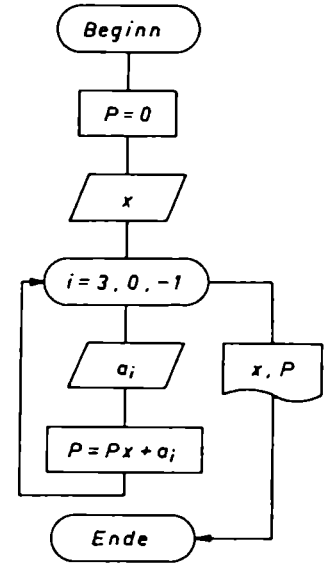
Einfach indizierte Variablenfelder bezeichnet man als *Vektoren.*

Zweifach indizierte Variablenfelder (mit Indizes $\geqq 1$) bezeichnet man als *Matrizen*[1].

Beachte: Die für indizierte Variablen reservierten Speicherplätze verkleinern den für das Objektprogramm zur Verfügung stehenden Gesamtplatz[2].
Aus Übersichtlichkeitsgründen stellen wir DIM-Vereinbarungen an den Anfang des Programms.
Durch explizite Vereinbarung von Feldern, deren Indizes $\leqq 10$ sind, kann man Speicherplatz sparen!
Die Verarbeitung von indizierten Variablen erfordert mehr Rechenzeit als die Verarbeitung von nichtindizierten Variablen.

BEISPIELE

1. Es ist ein Programm zur Berechnung des Werts von $P = a_3x^3 + a_2x^2 + a_1x + a_0$ unter Verwendung des Horner-Schemas zu schreiben.



Lösung: $P = [(a_3x + a_2)x + a_1]x + a_0$

Programm (für $P = -2x^3 + 5x^2 + x - 6$):

```
100 READ X
110 FOR I = 3 TO 0 STEP -1
120 READ A(I)
130 P = P*X+A(I)
140 NEXT I
150 PRINT X,P
160 DATA 3, -2,5,1,-6
```

Ausgabe:

```
 3                -12
```

Dieses Programm kann man auch ohne indizierte Variablen schreiben und damit kürzere Rechenzeiten erreichen, z. B.:

```
100 READ X
110 FOR I = 3 TO 0 STEP -1
120 READ A
130 P= P*X+A
140 NEXT I
150 PRINT X,P
160 DATA 3, -2,5,1,-6
```

Beachte:
DATA x, a_3, a_2, a_1, a_0

[1] In BASIC gibt es äußerst elegante und wirksame Anweisungen für das Operieren mit Matrizen. Diese werden im Abschnitt 11 ausführlich behandelt.

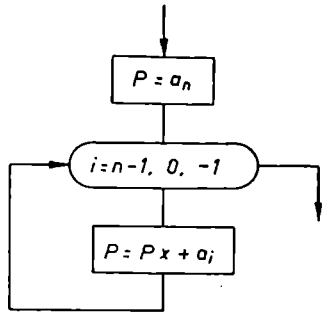
[2] Wenn 2074 Speicherplätze für indizierte Variablen reserviert werden, so wird die Länge des Objektprogramms auf 0 reduziert!

2. Es ist ein Programm für die Berechnung des Werts eines Polynoms n-ten Grades zu erstellen und mit den Polynomen

a) $x^5+x^3-2x^2+x+5$

b) $x^7-x^6-2x^3+x^2+5x+1$

im ersten Fall mit $x=2$ und im zweiten Fall mit $x=-1$ zu testen.



Programm[1]:

```
100:     A(N)*X↑N+...+A(1)*X+A(0)
110 READ N
120 PRINT
130 PRINT "GRAD N =";N
140 PRINT "A(0),...,A(N) =",
150 FOR I = 0 TO N
160 READ A(I)
170 PRINT A(I),
180 NEXT I
190 READ X
200 P = A(N)
210 FOR I = N-1 TO 0 STEP -1
220 P = P*X+A(I)
230 NEXT I
240 PRINT
250 PRINT "X =";X,"P(X) =";P
260 GOTO 110
270 DATA 5, 5,1,-2,1,0,1, 2
280 DATA 7, 1,5,1,-2,0,0,-1,1, -1
```

Ausgabe:

```
GRAD N = 5
A(0),...,A(N) = 5              1              -2              1
 0             1
X = 2          P(X) = 39

GRAD N = 7
A(0),...,A(N) = 1              5               1             -2
 0             0              -1               1
X =-1          P(X) =-3

OUT OF DATA  IN 110
```

[1] Wir nehmen willkürlich an: $n \leq 10$

3. Wir bezeichnen die Meßwerte einer Stichprobe mit $x_1, x_2, x_3, \ldots$ und die zugehörigen absoluten Häufigkeiten mit $f_1, f_2, f_3, \ldots$
Es ist ein Programm zur Berechnung der *relativen Häufigkeiten* $h_1, h_2, h_3, \ldots$ zu erstellen.

Lösung:

Es gilt: $h_j = \frac{f_j}{n}$ mit $n = f_1 + f_2 + \ldots + f_k = \sum_{j=1}^{k} f_j$

Um das lästige Zählen der k Klassen zu vermeiden, beenden wir das Programm mit Hilfe der „Schlußdaten" 1.11E11 und 1. ...[1]

Programm:

```
100 DIM X(50),F(50)
110 PRINT "  X(J)","F(J)","  H(J)"
120 FOR J = 1 TO 50
130 READ X(J),F(J)
140 IF X(J) = 1.11E11 THEN 170
150 N = N+F(J)
160 NEXT J
170 FOR I = 1 TO J-1
180 PRINT X(I),F(I),F(I)/N
190 NEXT I
200 PRINT "KLASSEN =";J-1,"ANZAHL =";N
210 DATA 15,1, 17,3, 19,8, 21,5, 23,2
220 DATA 1.11E11,1
```

Ausgabe:

```
 X(J)          F(J)          H(J)
 15            1             5.26316E-02
 17            3             .157895
 19            8             .421053
 21            5             .263158
 23            2             .105263
KLASSEN =  5    ANZAHL =  19
```

ÜBUNGEN

9.01 Welche indizierten Variablen sind nicht zulässig?

a) A(X ↑ 2) b) Z(Z) c) Y(−2)
d) C(N+1, N−1) e) A(M; N−5) f) R(3, 2, 5)

9.02 Welche Fehler enthalten die folgenden DIM-Anweisungen?

a) DIM A(5); B(6, 3) b) DIM C(20, 10, 20)
c) DIM D(5+3) d) DIM A(100, 100)

[1] Für 1 kann jede beliebige Zahl stehen.

9.03 Zeichne einen Programmablaufplan für das Lesen und Drucken des Vektors $(a_1, a_2, \ldots, a_n)$.

9.04 Zeichne je einen Programmablaufplan für spaltenweises[1] und zeilenweises[2] Lesen einer m,n-Matrix.

9.05 Es ist ein Programm zu schreiben für die Annäherung einer empirischen Verteilung $(x_1; f_1), (x_2; f_2), \ldots, (x_k; f_k)$ durch eine Normalverteilung[3].

Anleitung:

$$n = \Sigma f_j; \quad \bar{x} = \frac{\Sigma f_j x_j}{n}; \quad s^2 = \frac{1}{n-1}\left[\Sigma f_j x_j^2 - \frac{(\Sigma f_j x_j)^2}{n}\right]$$

Absolute Häufigkeit der j-ten Klasse: $f^* = \frac{nb}{s}\,\varphi(u_j)$ mit

$$u_j = \frac{x_j - \bar{x}}{s} \quad \text{und} \quad \varphi(u_j) = \frac{1}{\sqrt{2\pi}}\,e^{-\frac{1}{2}u_j^2}$$

Ausgabe:

X(J)	F(J)	U	PHI	F *
827.000	5	-1.59	.11	4.13
828.000	7	-.88	.27	9.93
829.000	24	-.18	.39	14.45
830.000	9	.53	.35	12.75
831.000	2	1.24	.19	6.82
832.000	4	1.95	.06	2.21
833.000	0	2.65	.01	.43
834.000	1	3.36	.00	.05

K = 8 N = 52 M = 829.25 S = 1.41291

9.06 Es ist ein Programm für das Ausdrucken der Tabelle der *Binomialkoeffizienten* (Seite 84) unter Ausnützung der Beziehung

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1} \quad \text{zu erstellen.}$$

---

[1] $a_{11}, a_{21}, \ldots, a_{m1}, a_{12}, a_{22}, \ldots, a_{m2}, \ldots, a_{1n}, a_{2n}, \ldots, a_{mn}$

[2] $a_{11}, a_{12}, \ldots, a_{1n}, a_{21}, a_{22}, \ldots, a_{2n}, \ldots, a_{m1}, a_{m2}, \ldots, a_{mn}$

[3] Es gelte: $b = x_2 - x_1 = x_3 - x_2 = \ldots\ldots$

10. ZEICHNEN VON FUNKTIONSKURVEN[1]

Lernziele

Den Aufbau und die Wirkung der TAB-Funktion angeben können.

Die Anwendbarkeit der TAB-Funktion angeben können.

Den (angenäherten) Graphen einer Funktion programmieren können.

In BASIC gibt es eine Funktion, nämlich die TAB-Funktion, mit deren Hilfe man den angenäherten Verlauf von Funktionskurven festlegen kann[2].

```
PRINT  TAB(5Ø); "+"
```

bewirkt, daß an der Stelle 5Ø der 72stelligen Zeile das Symbol + gedruckt wird.

```
PRINT  TAB(3Ø.5+2Ø*Y); "*"
```

bewirkt, daß an der Stelle 3Ø.5+2Ø*Y, wobei Y ein errechneter oder vorgegebener Wert ist[3], das Symbol * gedruckt wird.

Der Schreibkopf des Fernschreibers wird durch den Funktionsaufruf

TAB(T)

an die n-te Stelle gesetzt, wobei $n = \text{INT(T)}$ gilt.

Für $n = 72, 73, 74$ wird der Schreibkopf in Position 71 gebracht.

Für $n \geqq 75$ wird n zunächst durch 75 dividiert. Der Divisionsrest gilt dann als Argument.

Bei T= 12,6 wird der Schreibkopf in Position 12 gebracht.

Bei T= 85 wird der Schreibkopf in Position 10 gebracht.

Bei T=778 wird der Schreibkopf in Position 28 gebracht.

Bei T=$\leqq$Ø wird die TAB-Funktion ignoriert.

[1] Das Zeichnen von Kurven wird im Band BASIC-Praxis ausführlich behandelt:
Zeichnen einer Kurve
Zeichnen einer Funktion unter Verwendung logarithmischer Teilung an Abszisse und Ordinate
Zeichnen mehrerer Kurven
Zeichnen in Polarkoordinaten

[2] tab = tabulator [3] Annahme: $|y| \leq 1$

Der Schreibkopf kann durch den Funktionsaufruf TAB(T) nicht in derselben Zeile nach links gebracht werden.

Die TAB-Funktion darf nur in PRINT-Anweisungen, jedoch nicht in PRINTUSING-Anweisungen verwendet werden.

Bei der Darstellung von Funktionsgraphen legen wir einfachheitshalber die x-Achse von oben nach unten und die y-Achse von links nach rechts.

Wenn man eine maßstabgerechte Darstellung wünscht, so muß man den y-Wert mit 5/3 multiplizieren[1].

BEISPIELE

1. Es ist ein Programm für das Zeichnen der Sinuskurve im Intervall $0 \leq x \leq 2.0$ für die Schrittwerte $\Delta x = 0{,}2$ zu erstellen.

Programm:

```
100 PRINT "          0          0.5          1"
110 PRINT "          ↑ . . . . . ↑ . . . . . ↑"
120 FOR X = 0 TO 2 STEP .2
130 PRINT X;TAB(10.5+20*SIN(X));"*"
150 NEXT X
160 PRINT "          ↑ . . . . . ↑ . . . . . ↑"
170 PRINT "          0          0.5          1"
```

Ausgabe:

```
            0          0.5          1
            ↑ . . . . . ↑ . . . . . ↑
 0          *
 .2              *
 .4                   *
 .6                        *
 .8                             *
 1.                                  *
 1.2                                    *
 1.4                                     *
 1.6                                     *
 1.8                                    *
 2.                                    *
            ↑ . . . . . ↑ . . . . . ↑
            0          0.5          1
```

[1] Die Breite von 5 Druckpositionen ist gleich der Höhe von 3 Zeilen.

2. Es ist ein Programm für das Zeichnen der Sinus- und der Kosinuskurve zu erstellen.

Wir programmieren zunächst:

```
100 PRINT "          0          0.5          1"
110 PRINT "          ↑ . . . . ↑ . . . . ↑"
120 FOR X = 0 TO 2 STEP .2
130 PRINT X;TAB(9);"-";
140 PRINT TAB(10.5+20*SIN(X));"*";
145 PRINT TAB(10.5+20*COS(X));"+";TAB(31);"-"
150 NEXT X
160 PRINT "          ↑ . . . . ↑ . . . . ↑"
170 PRINT "          0          0.5          1"
```

und erhalten die Ausgabe[1]:

```
          0          0.5          1
          ↑ . . . . ↑ . . . . ↑
0         -*                    +-
.2        -    *                +-
.4        -        *          +  -
.6        -            *     +   -
.8        -                *+     -
1.        -                  *+   -
1.2       -                    *+-
1.4       -                     *+-
1.6       -                     *+-
1.8       -                    *+-
2.        -                   *+  -
          ↑ . . . . ↑ . . . . ↑
          0          0.5          1
```

Die Ursache der mißratenen „Kosinuskurve" ist leicht zu erkennen. Der Fernschreiber druckt befehlsgemäß zunächst * für $\sin x$ und erhält dann die Anweisung „Drucke + für $\cos x$". Beim TAB-Befehl kann sich der Schreibkopf nur nach rechts bewegen. Es wird somit bei $\cos x < \sin x$ zunächst * und an die folgende Druckstelle + geschrieben. Wir müssen diese Tatsache beim Programmieren berücksichtigen.

[1] Zum bequemen Zeichnen eines Gitters wurden an den Rändern Zeichen für das Anlegen eines Lineals gedruckt.

Unser Programm lautet nun:

```
100 PRINT " X        -1        -0.5        0         0.5        1"
110 PRINT "           ↑ . . . . ↑ . . . . ↑ . . . . ↑ . . . . ↑"
120 DEF FNT(X) = INT(30.5+20*X)
130 FOR X= 0 TO 3 STEP .2
140 PRINT X;TAB(9);"-";
150 Y=FNT(SIN(X))
160 Z=FNT(COS(X))
170 IF Z<Y THEN 210
180 IF Z=Y THEN 230
190 PRINT TAB(Y);"*";TAB(Z);"+";
200 GOTO 240
210 PRINT TAB(Z);"+";TAB(Y);"*";
220 GOTO 240
230 PRINT TAB(Y);"=";
240 PRINT TAB(51);"-"
250 NEXT X
260 PRINT "           ↑ . . . . ↑ . . . . ↑ . . . . ↑ . . . . ↑"
270 PRINT " X        -1        -0.5        0         0.5        1"
```

Nach der Berechnung von $y = \sin x$ und $z = \cos x$ und dem Schreiben dieser Worte bauen wir die bedingte Sprunganweisung 170 ein.

Wir fragen also, ob $\cos x < \sin x$ ist[1].

Wenn $\cos x < \sin x$ ist, so setzen wir bei Zeile 210 fort; wir drucken zuerst $+$, das Zeichen für den Kosinuswert, und dann das Zeichen $*$ für den Sinuswert.

Wenn $\cos x > \sin x$ ist, so wird die Anweisung 180 überlaufen und die Anweisung 190 ausgeführt, es wird zunächst $*$ für $\sin x$ und dann $+$ für $\cos x$ gedruckt.

Wenn $\cos x = \sin x$ ist, so wird von 170 zu 180 und von dort zu 230 gesprungen und laut dieser Anweisung das Zeichen $*$ einmal ausgedruckt.

Ausgabe:

```
X        -1        -0.5        0         0.5        1
          ↑ . . . . ↑ . . . . ↑ . . . . ↑ . . . . ↑
0        -                    *                    +-
.2       -                         *               +-
.4       -                              *         +  -
.6       -                                 *     +   -
.8       -                                    *      -
1.       -                                 *     *   -
1.2      -                              +           * -
1.4      -                         +                *-
1.6      -                    +                     *-
1.8      -                 +                       * -
2.       -             +                          *  -
2.2      -         +                             *    -
2.4      -     +                                *      -
2.6      -   +                             *          -
2.8      - +                            *             -
3.       -+                        *                  -
          ↑ . . . . ↑ . . . . ↑ . . . . ↑ . . . . ↑
X        -1        -0.5        0         0.5        1
```

[1] Unter Berücksichtigung der Rundung!

11. MATRIZENOPERATIONEN ...[1]

Lernziele

Den Begriff *Matrix* erklären können.

Die Anweisung für die Speicherreservierung für Matrizen angeben können.

Die Anweisungen für das Lesen und Ausdrucken von Matrizen angeben können.

Die Neudimensionierung einer Matrix erklären können.

Die Anweisung für

die Addition zweier Matrizen,
die Subtraktion zweier Matrizen,
die Multiplikation einer Matrix mit einer Zahl,
die Multiplikation zweier Matrizen,
die Berechnung der transponierten Matrix,
die Berechnung der Einheitsmatrix, Nullmatrix und konstanten Matrix,
die Berechnung der inversen Matrix

angeben können.

11.1. BEGRIFF DER MATRIX

Man bezeichnet das Zahlenschema $\begin{pmatrix} 1 & 0 & 5 \\ 3 & 2 & 1 \end{pmatrix}$ als 2, 3-*Matrix*. Es besteht aus zwei Zeilen, nämlich aus 1 0 5 und 3 2 1, und aus drei Spalten, nämlich aus $\begin{matrix} 1 \\ 3 \end{matrix}$ und $\begin{matrix} 0 \\ 2 \end{matrix}$ und $\begin{matrix} 5 \\ 1 \end{matrix}$.

(6 4 3 10 1) heißt 1, 5-Matrix oder *Zeilenvektor*.

$\begin{pmatrix} 4 \\ 3 \\ 9 \end{pmatrix}$ heißt 3, 1-Matrix oder *Spaltenvektor*.

Definition:

Ein rechteckiges Schema von reellen Zahlen, die in m Zeilen und in n Spalten angeordnet sind, bezeichnet man als m,n-*Matrix*.

[1] Für diesen Abschnitt werden Grundkenntnisse der Matrizenrechnung vorausgesetzt.

Es ist folgendes zu beachten:

Jede Matrix muß in BASIC durch genau einen Buchstaben gekennzeichnet sein.

Jede Matrix muß in BASIC durch eine Dimensionsanweisung DIM ... definiert sein, damit der benötigte Speicherbereich reserviert wird.
Dies gilt auch für Indizes $\leqq 10$.

Durch DIM A(M, N) werden (M+1)*(N+1) Speicherplätze reserviert, nämlich A(0, 0), A(0, 1), ..., A(0, N), ..., A(M, 0), A(M, 1), ..., A(M,N).

In BASIC beginnt jede Matrizenoperation mit MAT.

MAT-Anweisungen wirken erst ab Index 1.

Alle Elemente einer Matrix, auch die mit Index 0, sind durch andere Anweisungen aufrufbar.

11.2. LESEN UND DRUCKEN VON MATRIZEN

MAT READ bewirkt das Laden einer Matrix mit Daten.

MAT PRINT bewirkt das Schreiben einer Matrix.

BEISPIELE

1. Wenn man 15 Zahlen (1, 4, 23, 9, 0, −5, 3, 4, −9, 41, 12, 2, 5, −19, 15) einlesen, speichern und ausdrucken will, so kann man dies mittels einer FOR-Schleife, also mit

```
10 DIM A(15)
20 FOR I = 1 TO 15
30 READ A(I)
40 PRINT A(I),
50 NEXT I
60 DATA 1,4,23,9,0,-5,3,4,-9,41,12,2,5,-19,15
```

Beachte: Das 15stellige A-Feld umfaßt A(1), A(2), ..., A(15).

oder mit Hilfe der Matrizenanweisungen:

```
10 DIM A(1,15)
20 MAT READ A
30 MAT PRINT A
40 DATA 1,4,23,9,0,-5,3,4,-9,41,12,2,5,-19,15
```

Jedes dieser Programme bewirkt folgenden Output:

```
 1          4          23          9          0
-5          3          4          -9          41
 12         2          5          -19         15
```

Wenn wir die Zeile 1Ø durch 1Ø DIM A(15) ersetzen, so gibt uns der Rechner DIMENSION ERROR IN 2Ø bekannt.

Wir erkennen, daß ein n-spaltiger Zeilenvektor V bei Anwendung der Matrizenrechnung durch .. DIM V(1, n) definiert sein muß.

2. Die 3, 2-Matrix $\begin{pmatrix} 1 & 0 \\ 5 & 3 \\ 2 & 1 \end{pmatrix}$ ist einzulesen, zu speichern und auszudrucken.

Lösung:

```
1Ø DIM C(3,2)
2Ø MAT READ C
3Ø MAT PRINT C
4Ø DATA 1,Ø,5,3,2,1
```

```
 1   Ø
 5   3
 2   1
```

Beachte: Die Elemente der DATA-Zeile(n) werden zeilenweise in die Matrix gelesen!

3. Wir ändern die Zeilenanzahl des letzten Beispiels von 3 in 5; wir befehlen also: 1Ø DIM C(5, 2)
und erhalten den Output: OUT OF DATA IN 2Ø

Wir erkennen: Teilbereiche einer durch DIM festgelegten Matrix können mit MAT-Anweisungen nicht gelesen oder gedruckt werden!

4. Wenn wir das Programm des 2. Beispiels mit den Zeilen

```
35 MAT READ C(1, 5)
36 MAT PRINT C
45 DATA 1, 2, 3, 4, 5
```

erweitern, so erhalten wir folgenden Output:

```
 1   Ø
 5   3
 2   1

 1   2   3   4   5
```

Beachte: Die Matrix C wird durch die Anweisung

35 MAT READ C(6, 1)

neu dimensioniert!

Eine durch DIM A(M, N) dimensionierte Matrix A kann durch MAT READ A(I, J) neu dimensioniert werden, falls

$$(I+1)*(J+1) \leqq (M+1)*(N+1)$$

gilt[1].

11.3. GLEICHHEIT, ADDITION UND SUBTRAKTION VON MATRIZEN

Mit Matrizen kann man nach festgelegten Verknüpfungsregeln rechnen.

Zwei Matrizen A und B sind gleich, wenn sie in allen Elementen übereinstimmen.

Die Anweisung

```
MAT B = A
```

bewirkt, daß jedes Element von B mit dem entsprechenden Element von A belegt wird.

Voraussetzung: A und B müssen gleichartig[2] sein!

Beachte: Die Mehrfachanweisung MAT B = A = X ist nicht erlaubt.

Für gleichartige Matrizen[2] sind die Addition und die Subtraktion folgendermaßen definiert:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \pm \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{pmatrix} =$$

$$= \begin{pmatrix} a_{11} \pm b_{11} & a_{12} \pm b_{12} & \dots & a_{1n} \pm b_{1n} \\ a_{21} \pm b_{21} & a_{22} \pm b_{22} & \dots & a_{2n} \pm b_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} \pm b_{m1} & a_{m2} \pm b_{m2} & \dots & a_{mn} \pm b_{mn} \end{pmatrix}$$

In BASIC lauten die entsprechenden Anweisungen:

```
MAT S = A+B
MAT D = A-B
```

[1] M, N, I, J ... stehen hier für natürliche Zahlen.
Beachte: Durch MATPRINT kann keine Dimensionierung vorgenommen werden.

[2] gleiche Anzahl von Zeilen und von Spalten

11.4. MULTIPLIKATION EINER MATRIX MIT EINER ZAHL

Die Multiplikation einer Matrix mit einer Zahl ist folgendermaßen definiert:

$$k \cdot \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} = \begin{pmatrix} ka_{11} & \dots & ka_{1n} \\ \dots & \dots & \dots \\ ka_{m1} & \dots & ka_{mn} \end{pmatrix}$$

In BASIC lauten die entsprechenden Anweisungen[1]:

```
10 DIM A(10,2), B(10,2)
20 READ K
30 MAT READ A
40 MAT B = (K)*A
50 MAT PRINT B
60 DATA .........
```

Allgemein lautet die BASIC-Anweisung für die Multiplikation eines Terms **T** mit einer Matrix A:

```
MAT B=(T)*A
```

11.5. TRANSPONIERTE MATRIX

Wenn man in einer Matrix $\mathfrak{A}$ die Zeilen und Spalten vertauscht, so erhält man die *transponierte Matrix* $\mathfrak{A}^T$.

Beispiele: $\mathfrak{A} = \begin{pmatrix} 5 & 7 \\ 1 & 2 \end{pmatrix}$ $\qquad \mathfrak{A}^T = \begin{pmatrix} 5 & 1 \\ 7 & 2 \end{pmatrix}$

$\mathfrak{D} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ $\qquad \mathfrak{D}^T = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$

Die entsprechende BASIC-Anweisung lautet[2]:

```
MAT B=TRN(A)
```

Beachte: A und B müssen entsprechende Dimension haben, z. B.:

```
50 DIM A(5, 2), B(2, 5)
```

MAT A=TRN(A) ist nicht erlaubt!

---

[1] Wir nehmen eine 10, 2-Matrix an!

[2] Wir bezeichnen hier die Matrix mit A und ihre Transponierte mit B.

11.6. EINHEITSMATRIX

In der Matrizenrechnung ist die *Einheitsmatrix* besonders wichtig.

Darunter versteht man eine quadratische Matrix (Anzahl der Zeilen = Anzahl der Spalten), bei der alle Glieder der Hauptdiagonale gleich 1 sind und alle übrigen Glieder gleich 0.

Beispiele:

$$\mathfrak{E} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \mathfrak{E} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad \mathfrak{E} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

In BASIC kann man eine Einheitsmatrix durch die Anweisung

MAT A=IDN(T_1, T_1) ...[1]

definieren.

Durch die Anweisung

MAT A=IDN

wird die quadratische n,n-Matrix A durch die n,n-Einheitsmatrix ersetzt.

BEISPIELE

1.
```
10 DIM A(20,20)
20 MAT READ A
.. ................
80 MAT A = IDN(4,4)
```

speichert die Einheitsmatrix $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

Beachte: Die 20,20-Matrix A wurde durch

80 MAT A=IDN(4, 4)

neu dimensioniert.

---

[1] identity ≙ Identität

2. Es sind die Matrizen (5), $\begin{pmatrix} 1 & 1 \\ 2 & 2 \end{pmatrix}$ und $\begin{pmatrix} 3 & 3 & 3 \\ 4 & 4 & 4 \\ 5 & 5 & 5 \end{pmatrix}$ durch Einheitsmatrizen zu ersetzen und diese auszudrucken.

Programm:

```
10 DIM A(1,1),B(2,2),C(3,3)
20 MAT READ A,B,C
30 MAT PRINT A,B,C
40 MAT A = IDN
50 MAT B = IDN
60 MAT C = IDN
70 MAT PRINT A,B,C
80 DATA 5, 1,1,2,2, 3,3,3,4,4,4,5,5,5
```

Ausgabe:

```
 5

 1             1

 2             2

 3             3             3

 4             4             4

 5             5             5

 1

 1             0

 0             1

 1             0             0

 0             1             0

 0             0             1
```

11.7. NULLMATRIX, KONSTANTE MATRIX

Eine Matrix, deren sämtliche Elemente 0 sind, heißt ohne Rücksicht auf ihren Typ *Nullmatrix*.

Eine Matrix, deren sämtliche Elemente 1 sind, heißt *konstante Matrix*.

Durch die Anweisung

```
MAT  A=ZER
```

werden alle Elemente der Matrix A mit 0 belegt. ...[1]

Durch die Anweisung

```
MAT  A=CON
```

werden alle Elemente der Matrix A mit 1 belegt. ...[2]

Als Spezialfall für die Erzeugung einer Matrix mit neuen Dimensionen lauten diese Anweisungen:

```
MAT  Z=ZER(T1, T2)
```

Alle Elemente der Matrix Z werden mit $\emptyset$ belegt.

und

```
MAT  C=CON(T1, T2)
```

Alle Elemente der Matrix C werden mit 1 belegt.

In beiden Fällen bestimmen die Werte von T_1 und T_2 die Dimension der Matrix.

Beachte: Vor Beginn einer Rechnung muß für jede Matrix der maximal benötigte Platz durch eine DIM-Anweisung reserviert werden.

Durch die Anweisungen

```
      ZER
oder  IDN
oder  CON
```

kann eine Matrix während des Programmablaufs verkleinert werden. Dabei gehen jedoch die bisher gespeicherten Werte des Feldes verloren.

Die neue Matrix darf selbstverständlich nicht mehr Platz in Anspruch nehmen, als durch die DIM-Anweisung reserviert wurde.

[1] zero ≙ Null

[2] constant ≙ Konstante

BEISPIELE

1.
```
10 DIM A(5,5), B(5,5), C(20,20)
20 MAT READ A,B
.. ..............................
130 MAT C = ZER(5,5)
140 MAT C = A-B
... ..............................
```

Beachte: Die Anweisung 130 MAT C = ZER(5, 5) macht von der Reservierung der 20,20-Matrix C nur teilweisen Gebrauch und erzeugt eine 5,5-Nullmatrix.

Durch ähnliche Anweisungen können die Matrizen A, B, C im späteren Programmablauf neu dimensioniert werden.

2.
```
10 DIM A(1,10), B(1,10), C(1,10)
.. ..............................
80 READ N
90 MAT A = ZER(1,N)
100 MAT B = ZER(1,N)
110 MAT C = ZER(1,N)
120 MAT READ A,B
130 MAT C = A+B
... ..............................
```

Beachte: Durch die Anweisungen 90, 100, 110 werden die Vektoren A, B, C (Nullvektoren der Dimension N) erzeugt.

11.8. MULTIPLIKATION VON MATRIZEN

Die Multiplikation zweier Matrizen $\mathfrak{A}$ und $\mathfrak{B}$ ist folgendermaßen definiert:

Ist $\mathfrak{A}$ eine m,n-Matrix und $\mathfrak{B}$ eine n,p-Matrix, so ist $\mathfrak{A} \cdot \mathfrak{B} = \mathfrak{C}$ eine m,p-Matrix, für deren Elemente c_{ik} gilt: c_{ik} ist das skalare Produkt der i-ten Zeile von $\mathfrak{A}$ mit der k-ten Spalte von $\mathfrak{B}$.

Beispiel:

$$\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \cdot \begin{pmatrix} 0 & 2 \\ 3 & 1 \end{pmatrix} = \begin{pmatrix} 1 \cdot 0 + 3 \cdot 3 & 1 \cdot 2 + 3 \cdot 1 \\ 2 \cdot 0 + 4 \cdot 3 & 2 \cdot 2 + 4 \cdot 1 \end{pmatrix} = \begin{pmatrix} 9 & 5 \\ 12 & 8 \end{pmatrix}$$

MAT C=A*B

Testprogramm:

```
10 DIM A(2,2), B(2,2), C(2,2)
20 MAT READ A,B
30 MAT C = A*B
40 MAT PRINT C
50 DATA 1,3,2,4, 0,2,3,1
```

Ausgabe:

```
 9          5

 12         8
```

11.9. INVERSE MATRIX

Wichtig ist noch die Berechnung der Kehrmatrix oder inversen Matrix einer gegebenen quadratischen Matrix.

Es gilt:

$\mathfrak{A}$ ist eine quadratische nichtsinguläre Matrix. ...[1]

$\mathfrak{A}^{-1}$ heißt *Kehrmatrix* oder *inverse Matrix* von $\mathfrak{A}$ genau dann, wenn gilt:

$$\mathfrak{A} \cdot \mathfrak{A}^{-1} = \mathfrak{A}^{-1} \cdot \mathfrak{A} = \mathfrak{E}$$

Die Berechnung einer Kehrmatrix ist sehr zeitraubend. In BASIC haben wir es leichter. Die Anweisung, die Inverse von $\mathfrak{A}$ zu bilden und im Feld $\mathfrak{B}$ zu speichern, lautet:

MAT B=INV(A)

Beachte: Beide Matrizen müssen gleich dimensioniert sein.

MAT A=INV(A) ist nicht erlaubt!

Ist eine Matrix singulär, d. h. ihre Determinante gleich Null, so erfolgt die Warnmeldung

NEARLY SINGULAR MATRIX

[1] Eine Matrix $\mathfrak{A}$ heißt *nichtsingulär*, wenn $\det(\mathfrak{A}) \neq 0$ ist.

BEISPIELE

1. Gegeben sei die Matrix $\mathfrak{A} = \begin{pmatrix} 5 & 6 & 3 \\ 7 & 8 & 2 \\ 0 & 5 & 0 \end{pmatrix}$

Es ist ein Programm für die Berechnung von $\mathfrak{A} \cdot \mathfrak{A}^{-1}$ zu erstellen.

Programm:

```
100 DIM A(3,3), B(3,3), C(3,3)
110 MAT READ A
120 PRINT "MATRIX A"
130 MAT PRINT A
140 PRINT "INV(A)"
150 MAT B = INV(A)
160 MAT C = A*B
170 MAT PRINT B
180 PRINT " A*INV(A)"
190 MAT PRINT C
200 DATA 5,6,3,  7,8,2,  0,5,0
```

Ausgabe:

```
MATRIX A
 5                6                3

 7                8                2

 0                5                0


INV(A)
-.181818          .272727         -.218182

-4.65661E-10      4.65661E-10      .2

 .636364         -.454545         -3.63636E-02


 A*INV(A)
 1.              -3.72529E-09     -2.21189E-09

-3.72529E-09      1.              -8.14907E-10

-2.32831E-09      2.32831E-09      1.
```

2. Es ist ein Programm für die Lösung eines linearen Gleichungssystems in n Variablen zu erstellen, wobei $n \leqq 20$ vorausgesetzt wird.

Als Testbeispiel ist das System

$$\begin{aligned} -6x_1 + 7x_2 + 5x_3 - 6x_4 &= -35 \\ 5x_1 - 9x_2 + 12x_3 - 7x_4 &= 66 \\ 4x_1 + 2x_2 - 3x_3 + 2x_4 &= -22 \\ -7x_1 - 3x_2 + 4x_3 - 5x_4 &= 18 \end{aligned}$$

zu verwenden.

Lösung:

Das Gleichungssystem

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= y_1 \\ &\ldots\ldots\ldots \\ &\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= y_n \end{aligned}$$

kann im Matrizenkalkül bei Beachtung der Multiplikationsvorschrift zweier Matrizen auch geschrieben werden:

$$\mathfrak{A} \cdot \mathfrak{X} = \mathfrak{Y}$$

mit $\mathfrak{A} = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \ldots & \ldots & \ldots \\ a_{n1} & \ldots & a_{nn} \end{pmatrix}$ sowie $\mathfrak{X} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ und $\mathfrak{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$

Beachte: Die Anwendung der Matrizenmultiplikation auf die n,n-Matrix $\mathfrak{A}$ und $n,1$-Matrix $\mathfrak{X}$ ergibt die $n,1$-Matrix $\mathfrak{Y}$.

Die Multiplikation beider Seiten der Matrizengleichung $\mathfrak{A} \cdot \mathfrak{X} = \mathfrak{Y}$ mit der inversen Matrix zu $\mathfrak{A}$, also mit $\mathfrak{A}^{-1}$, ergibt:

$\mathfrak{A}^{-1} \cdot \mathfrak{A} \cdot \mathfrak{X} = \mathfrak{A}^{-1} \cdot \mathfrak{Y}$

Nun ist $\mathfrak{A}^{-1} \cdot \mathfrak{A} = \mathfrak{E}$ die Einheitsmatrix.

Daher gilt: $\mathfrak{E} \cdot \mathfrak{X} = \mathfrak{A}^{-1} \cdot \mathfrak{Y}$

Wegen $\mathfrak{E} \cdot \mathfrak{X} = \mathfrak{X}$ erhalten wir: $\underline{\mathfrak{X} = \mathfrak{A}^{-1} \cdot \mathfrak{Y}}$

Die Operation MAT X=INV(A)*Y ist in BASIC nicht zugelassen; sie muß in 2 Zeilen aufgelöst werden, nämlich in:

```
MAT  B=INV(A)
MAT  X=B*Y
```

Programm:

```
100 DIM A(20,20), B(20,20), X(20,1), Y(20,1), Z(20,1)
110 READ N
120 MAT A = ZER(N,N)
130 MAT B = ZER(N,N)
140 MAT X = ZER(N,1)
150 MAT Y = ZER(N,1)
160 MAT Z = ZER(N,1)
170 MAT READ A, Y
180 MAT B = INV(A)
190 MAT X = B*Y
200 MAT Z = A*X
210 PRINT "KOEFFIZIENTENMATRIX"
220 MAT PRINT  A
230 PRINT "RECHTE SEITE","LOESUNGSVEKTOR","PROBE: A*X"
240 FOR I = 1 TO N
250 PRINT Y(I,1),X(I,1),Z(I,1)
260 NEXT I
270 DATA 4,  -6,7,5,-6, 5,-9,12,-7, 4,2,-3,2, -7,-3,4,-5
280 DATA -35,66,-22,18
```

Ausgabe:

```
KOEFFIZIENTENMATRIX
-6              7              5             -6

 5             -9             12             -7

 4              2             -3              2

-7             -3              4             -5

RECHTE SEITE   LOESUNGSVEKTOR PROBE: A*X
-35            -1.61089       -35
 66            -5.09339        66.
-22             6.28016       -22
 18             6.73541        18.



USED     2.67 UNITS.
```

3. Durch n Punkte $(x_1; y_1), (x_2; y_2), \ldots, (x_n; y_n)$ ist eine Parabel zweiter Ordnung so zu legen, daß die Summe der Ordinatenabweichungsquadrate möglichst klein wird.

Die Gleichung der *Ausgleichsparabel* lautet: $y = a_0 + a_1 x + a_2 x^2$

Die Koeffizienten a_0, a_1, a_2 findet man aus dem System der *„Gauß-Normalgleichungen"*:

$$\begin{aligned} n a_0 &+ [x]a_1 &+ [x^2]a_2 &= [y] \\ [x]a_0 &+ [x^2]a_1 &+ [x^3]a_2 &= [xy] \\ [x^2]a_0 &+ [x^3]a_1 &+ [x^4]a_2 &= [x^2y] \end{aligned}$$

Es bedeutet:
$[x] = \Sigma x$
$[x^2] = \Sigma x^2$ usw.

Setzen wir:

$N = C(1, 1)$; $[x] = C(1, 2) = C(2, 1)$;

$[x^2] = C(1, 3) = C(2, 2) = C(3, 1)$; $[x^3] = C(2, 3) = C(3, 2)$;

$[x^4] = C(4, 4)$; $[y] = L(1, 1)$; $[xy] = L(2, 1)$; $[x^2y] = L(3, 1)$

also $\begin{pmatrix} n & [x] & [x^2] \\ [x] & [x^2] & [x^3] \\ [x^2] & [x^3] & [x^4] \end{pmatrix} = \mathfrak{C}$ und $\begin{pmatrix} [y] \\ [xy] \\ [x^2y] \end{pmatrix} = \mathfrak{L}$

sowie für den Lösungsvektor $\mathfrak{A} = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix}$,

so können wir das Gauß-Normalgleichungssystem in Matrizenform schreiben:

$$\mathfrak{C} \cdot \mathfrak{A} = \mathfrak{L}$$

Durch Multiplikation mit $\mathfrak{C}^{-1}$ erhalten wir:

$\mathfrak{C}^{-1} \cdot \mathfrak{C} \cdot \mathfrak{A} = \mathfrak{C}^{-1} \cdot \mathfrak{L}$

Es ist $\mathfrak{C}^{-1} \cdot \mathfrak{C} = \mathfrak{E}$

und somit: $\mathfrak{E} \cdot \mathfrak{A} = \mathfrak{D} \cdot \mathfrak{L}$

bzw. $\underline{\mathfrak{A} = \mathfrak{D} \cdot \mathfrak{L}}$

$$\mathfrak{D} = \mathfrak{C}^{-1} = \begin{pmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{pmatrix}$$

Der *mittlere Fehler der Beobachtungen* y_i ist

$$F = \sqrt{\frac{Q}{n-3}}$$

mit $Q = [yy] - a_0[y] - a_1[xy] - a_2[x^2y]$ ...[1]

Für die *Fehler der errechneten Koeffizienten* gilt:

$F_{a0} = F \cdot \sqrt{d_{11}}$ $F_{a1} = F \cdot \sqrt{d_{22}}$ $F_{a2} = F \cdot \sqrt{d_{33}}$

[1] $n - 3$ ist der Grad der Überbestimmung. Durch 3 Punkte ist eine Parabel zweiter Ordnung bestimmt. Es liegen n Meßwerte vor; es bleiben daher $n - 3$ Kontrollmessungen.

Programm:

```
100:###   ##.########   ##.########
110:  I       X(I)          Y(I)
120:             MESSWERTE
130:  I       A(I)        F(A(I))
140:                  F =###.########
150 DIM C(3,3),D(3,3),A(3,1),L(3,1)
160 PRINTUSING 120
170 PRINTUSING 110
180 READ N
190 C(1,1) = N
200 FOR I = 1 TO N
210 READ X, Y
220 PRINTUSING 100,I,X,Y
230 C(1,2) = C(2,1) = C(1,2)+X
240 C(1,3) = C(3,1) = C(2,2) = C(1,3)+X*X
250 C(2,3) = C(3,2) = C(2,3)+X*3
260 C(3,3) = C(3,3)+X*4
270 L(1,1) = L(1,1)+Y
280 L(2,1) = L(2,1)+X*Y
290 L(3,1) = L(3,1)+Y*X*X
300 Q = Q+Y*Y
310 NEXT I
320 MAT D = INV(C)
330 MAT A = D*L
340 FOR I = 1 TO 3
350 Q = Q - A(I,1)*L(I,1)
360 NEXT I
370 F = SQR(Q/(N-3))
380 PRINT
390 PRINT "AUSGLEICHSPARABEL Y = A(0) + A(1)*X + A(2)*X↑2"
400 PRINTUSING 130
410 FOR I = 1 TO 3
420 II = I-1
430 FI = F*SQR(D(I,I))
440 PRINTUSING 100,II,A(I,1),FI
450 NEXT I
460 PRINT
470 PRINTUSING 140,F
480 DATA 6,  -2.35, 0.50, 3.68, 4.70, 6.76, 9.84
```

Ausgabe:

```
          MESSWERTE
  I       X(I)          Y(I)
  1   -2.000E+00    3.500E+01
  2     .000E+00    5.000E+01
  3    3.000E+00    6.800E+01
  4    4.000E+00    7.000E+01
  5    6.000E+00    7.600E+01
  6    9.000E+00    8.400E+01

AUSGLEICHSPARABEL Y = A(0) + A(1)*X + A(2)*X↑2
  I       A(I)        F(A(I))
  0    4.964E+01    8.186E-01
  1    6.467E+00    3.633E-01
  2   -3.024E-01    4.742E-02

               F =  1.472E+00
```

Die Gleichung der gesuchten Ausgleichsparabel lautet somit:

$y = 49{,}64 + 6{,}467\,x - 0{,}3024\,x^2$

12. UNTERPROGRAMME [1]

Lernziele

Die Begriffe *Unterprogramm*, *Hauptprogramm*, *Aktualparameter* und *Formalparameter* erklären können.

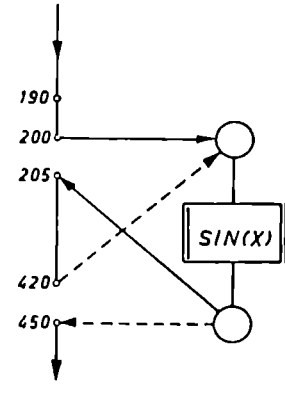
Die Wirkung der Anweisung RETURN erklären können.

Die Wirkung der GOSUB-Anweisung erklären können.

Die Wirkung der CALL-Anweisung erklären können.

Wir gehen von folgendem Programmteil aus:

```
190  ..............
200  W=SIN(2*A+.5)
205  ..............
...  ..............
...  ..............
420  K=SIN(B)
450  ..............
...  ..............
```



bewirkt den Sprung aus dem *Hauptprogramm* in das *Unterprogramm* (kurz *Upro*) „Berechnung von SIN(X)".

Im Upro wird ohne unser Zutun zunächst der Formalparameter X durch den momentanen Wert von 2*A+.5 ersetzt.

Dann wird SIN(2*A+.5) berechnet.

Dieser Wert wird an das Hauptprogramm übergeben und von diesem unter dem Namen W gespeichert.

Sodann wird im Hauptprogramm mit der nächsten Anweisung, also mit der Anweisung 205, fortgesetzt.

Durch die Anweisung 420 erfolgt wieder ein Sprung in das Upro „Berechnung von SIN(X)". Der Formalparameter X wird mit dem momentanen Wert von B belegt, SIN(B) berechnet, dieser Wert dem Hauptprogramm übergeben und anschließend im Hauptprogramm mit der Anweisung 450 fortgesetzt.

---

[1] Vor dem Studium dieses Abschnitts ist es günstig, den Abschnitt 5.2 zu wiederholen!

Außer den Standardfunktionen haben wir bereits eine andere Art von Unterprogrammen kennengelernt, die *vereinbarten Funktionen*.

Durch 100 DEF FNF(X)= ..... sei eine Funktion $f(x)$ vereinbart. Die Anweisung 350 Z=FNF(A) bewirkt dann den Sprung in das Upro „Berechnung von $f(x)$".

Im Upro wird der Formalparameter X durch den aktuellen Wert des Aktualparameters A ersetzt, $f(a)$ berechnet und an das Hauptprogramm übergeben. Dann wird mit der auf 350 Z=FNF(A) folgenden Anweisung des Hauptprogramms fortgesetzt.

Wir fassen zusammen:

Das Programm, mit dem die Rechnung beginnt, heißt *Hauptprogramm*.

Selbständige Programmteile, die zur Bewältigung von Teilaufgaben beliebig oft aufgerufen werden können, heißen *Unterprogramme*.

Wir haben bisher zwei Arten von Unterprogrammen besprochen:

1. Standardfunktionen.
2. Vereinbarte Funktionen.

Die Anweisung für eine vereinbarte Funktion darf bekanntlich nicht länger als eine Zeile sein. Wenn man Unterprogramme aufrufen will, die aus mehreren Anweisungen bestehen, verwendet man die GO SUB-Anweisung[1].

Die Anweisung

GOSUB n

bewirkt den Aufruf des mit der Zeilennummer n beginnenden Upros und den automatischen Rücksprung in das Hauptprogramm[2].

Beachte:

Jedes Unterprogramm endet mit der unbedingten Sprunganweisung

RETURN

Diese Anweisung bewirkt den Rücksprung auf die unmittelbar hinter GOSUB n stehende Anweisung[3].

---

[1] SUB ist eine Abkürzung für subprogram (≙ Unterprogramm).

[2] go sub ≙ geh zum Unterprogramm

[3] return ≙ zurückkehren

Ein Unterprogramm soll nur mittels einer GOSUB-Anweisung betreten und über eine RETURN-Anweisung verlassen werden.

Ein Unterprogramm kann mehrere RETURN-Anweisungen aufweisen (mehrere Ausgänge!). Die zuerst erreichte RETURN-Anweisung wird ausgeführt.

Unterprogramme können geschachtelt sein[1]: In einem Upro wird durch eine GOSUB-Anweisung ein anderes Upro aufgerufen usw.

Alle Variablen und vereinbarten Funktionen gelten gemeinsam für das Hauptprogramm und alle aufgerufenen Upros.

BEISPIELE

1. Den größten gemeinsamen Teiler zweier natürlicher Zahlen kann man mit Hilfe des Euklid-Algorithmus ermitteln. Wir erklären diesen zunächst anhand eines Zahlenbeispiels: Es ist der größte gemeinsame Teiler (ggT) der Zahlen 60 und 36 zu ermitteln.

Es gilt:
$$\begin{aligned} 60 &= 36 \cdot 1 + 24 \\ 36 &= 24 \cdot 1 + 12 \\ 24 &= 12 \cdot 2 + 0 \end{aligned} \qquad ggT(60, 36) = 12$$

Der letzte, nicht verschwindende Rest ist der gesuchte ggT.

Allgemein gilt:

$$\begin{aligned} a &= b \cdot q_1 + r_1 && (a > b) \\ b &= r_1 q_2 + r_2 \\ r_1 &= r_2 q_3 + r_3 \\ &\ldots\ldots\ldots\ldots\ldots && \underline{ggT = r_n} \\ r_{n-2} &= r_{n-1} q_n + r_n \\ r_{n-1} &= r_n \cdot q_{n+1} + 0 \end{aligned}$$

[1] Schachtelungen bis 39 Upros sind bei Mark I/7 zugelassen.

Wir erkennen die gemeinsame Struktur:

$$x = y \cdot q + r$$

bzw. $r = x - yq \quad \text{mit} \quad q = \text{INT}\left(\frac{x}{y}\right)$ ...[1]

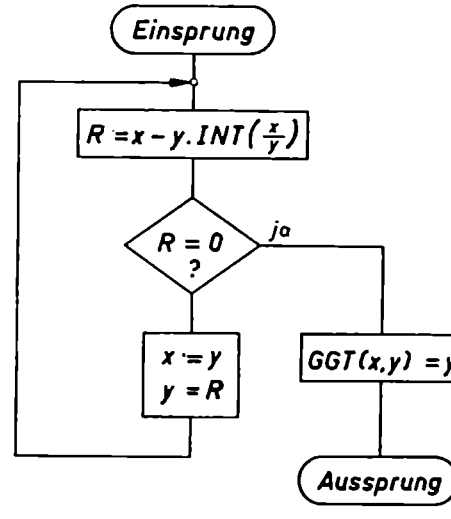
also $\underline{r = x - y \cdot \text{INT}\left(\frac{x}{y}\right)}$

Mit diesem Verfahren besitzen wir ein Unterprogramm für das Hauptprogramm „Ermittlung des größten gemeinsamen Teilers mehrerer natürlicher Zahlen". ...[2]

Zunächst halten wir den Programmablauf graphisch fest und überprüfen mit den Zahlen 12, 42, 18 den Programmablauf.

Hauptprogramm:
$x = 12; \quad y = 42$

Unterprogramm:
$r = 12 - 42 \cdot \text{INT}\left(\frac{12}{42}\right) =$
$= 12 - 42 \cdot 0 = 12$
$x = 42; \quad y = 12$
$r = 42 - 12 \cdot \text{INT}\left(\frac{42}{12}\right) =$
$= 42 - 12 \cdot 3 = 6$
$x = 12; \quad y = 6$
$r = 12 - 6 \cdot \text{INT}\left(\frac{12}{6}\right) =$
$= 12 - 6 \cdot 2 = 0$
$ggT(12, 42) = 6$



Hauptprogramm:

$x = 18; \quad y = 6$

.....

.....

[1] q ist die nächstkleinere ganze Zahl von $\frac{x}{y}$.

[2] Wir beschränken uns hier auf den ggT dreier natürlicher Zahlen.

Wir müssen noch den Fall $x=y$ überprüfen.

Wir wählen $x=y=5$; $z=75$

Hauptprogramm:

$x=5$; $y=5$

Unterprogramm:

$r=5-5 \cdot \mathrm{INT}\left(\frac{5}{5}\right)=5-5 \cdot 1=0$

$ggT(5, 5)=5$

.....

.....

Programm:

```
100 READ X,Y,Z
110 PRINT "DER GGT VON";X;Y;Z;"IST";
120 GOSUB 310
130 X = Z
140 GOSUB 310
150 PRINT Y
160 GOTO 100
170 DATA 12,42,18, 840,90,18900, 5,5,75
300:  UPRO GGT
310 R = X-Y*INT(X/Y)
320 IF R = 0 THEN 360
330 X = Y
340 Y = R
350 GOTO 310
360 RETURN
```

Ausgabe:

```
DER GGT VON 12    42    18   IST 6
DER GGT VON 840   90    18900   IST 30
DER GGT VON 5  5  75   IST 5

OUT OF DATA  IN 100
```

2. Ein Unterprogramm für die Berechnung von *Arkussinuswerten* lautet[1]:

```
90REM    UPRO ARCSIN(X)
100  IF ABS(X) = 1 THEN 130
110  Y = ATN(X/SQR(1-X↑2))
120  RETURN
130  Y = X*1.57079633
140  RETURN
```

Wir testen dieses Upro mit folgendem Hauptprogramm:

```
10  PRINT "SIN(X)","X(RAD)","X(GRAD)"
20  FOR X = -1 TO 1 STEP .5
30  GOSUB 100
40  PRINT X,Y,Y*57.2957795
50  NEXT X
...[2]  60  STOP
```

und erhalten:

SIN(X)	X(RAD)	X(GRAD)
-1	-1.5708	-90.
-0.5	-0.523599	-30.
0	0	0
0.5	0.523599	30.
1	1.5708	90.

3. Ein Unterprogramm für die Umrechnung von Radianten B in Altgrade G, Minuten M und Sekunden S lautet:

```
100:    UPRO WINUM
110  W = B*57.2957795
120  G = INT(W)
130  M1 = 60*(W-G)
140  M = INT(M1)
150  S = 60*(M1-M)
160  RETURN
```

Programmvariante:

```
110  W = B*2.06264806E5
120  G = INT(W/3600)
130  M = INT(W/60)-G*60
140  S = W-G*3600-M*60
150  RETURN
```

---

[1] Siehe Seite 52. [2] Diese Anweisung schirmt das Upro ab!

Wir testen dieses Upro mit den Winkeln $\widehat{\alpha_1}=0$, $\widehat{\alpha_2}=1$, $\widehat{\alpha_3}=\frac{\pi}{3}$ durch folgendes **Hauptprogramm**:

```
10 READ B
20 GOSUB 110
30 PRINT B, G, M, INT(10*S+.5)/10
40 GOTO 10
50 DATA 0, 1, 1.04719755
```

und erhalten die Ausgabe:

```
 0              0              0              0
 1              57             17             44.8
 1.0472         60             0              0
OUT OF DATA IN 10
```

Beachte: Für den praktischen Gebrauch entfernt man den Testteil 10—50 und speichert das Upro im Bibliotheksspeicher.

In jedem Hauptprogramm, das dieses Unterprogramm verwenden soll, müssen die Variablennamen mit den entsprechenden Variablennamen dieses Upros übereinstimmen.

4. Eine Gerade mit dem Richtvektor $\begin{pmatrix} r_1 \\ r_2 \end{pmatrix}$ geht durch den Punkt $P_0(x_0/y_0)$.

Es ist ein Unterprogramm für die Berechnung der Koeffizienten A, B, C der Geradengleichung $Ax+By+C=0$ zu erstellen.

Lösung:

Für $r_1 \neq 0$ gilt: $y-y_0=k(x-x_0)$ mit $k=\frac{r_2}{r_1}$

also: $kx-y+y_0-kx_0=0$ ...[1]

Für $r_1=0$ gilt: $x=x_0$, also $1\cdot x+0\cdot y-x_0=0$ ...[2]

Unser Upro lautet:

```
70  REM UPRO "PUNRI"
80  REM EINGABE: X0, Y0, R1, R2
90  REM AUSGABE: A, B, C
100 IF R1 = 0 THEN 150
110 A = R2/R1
120 B = -1
130 C = Y0-A*X0
140 RETURN
150 A = 1
160 B = 0
170 C = -X0
180 RETURN
```

[1] $A=k$; $B=-1$; $C=y_0-kx_0$
[2] $A=1$; $B=0$; $C=-x_0$

Nach einem Test speichern wir dieses Upro unter dem Namen PUNRI (eine von uns gewählte Abkürzung für *Punkt-Richtungsform*) durch den Systembefehl SAVE im Bibliotheksspeicher. ...[1]

Wir zeigen nun, wie man dieses Unterprogramm in ein Programm einbauen kann. Es sind der Punkt $P_5(x_5|y_5)$ und der Richtvektor $\begin{pmatrix} h_5 \\ v_5 \end{pmatrix}$ gegeben.

Die Koeffizienten der Geradengleichung sind zu berechnen.

Hauptprogramm für Test:

```
100 READ X5, Y5, H5, V5
110 X0 = X5
120 Y0 = Y5
130 R1 = H5
140 R2 = V5
150 CALL PUNRI
160 PRINT A,B,C
165 GOTO 100
170 DATA 0, 2, 1, 2
180 DATA 3, 0, 0, 5
```

Ausgabe:

```
 2                -1                2
 1                 0               -3

OUT OF DATA  IN 100
```

Beachte: Durch die Anweisung

```
CALL PUNRI
```

wird das externe Upro PUNRI aufgerufen und mit dem Hauptprogramm verarbeitet[1].

Durch CALL-Anweisungen können auch Upros der Systembibliothek aufgerufen werden!

[1] Abschnitt 14; save ≙ aufheben, speichern

[1] call ≙ rufen

13. WEITERE PROGRAMMIERUNGSBEISPIELE

Dieser Abschnitt dient zur Abrundung des erworbenen Wissens.

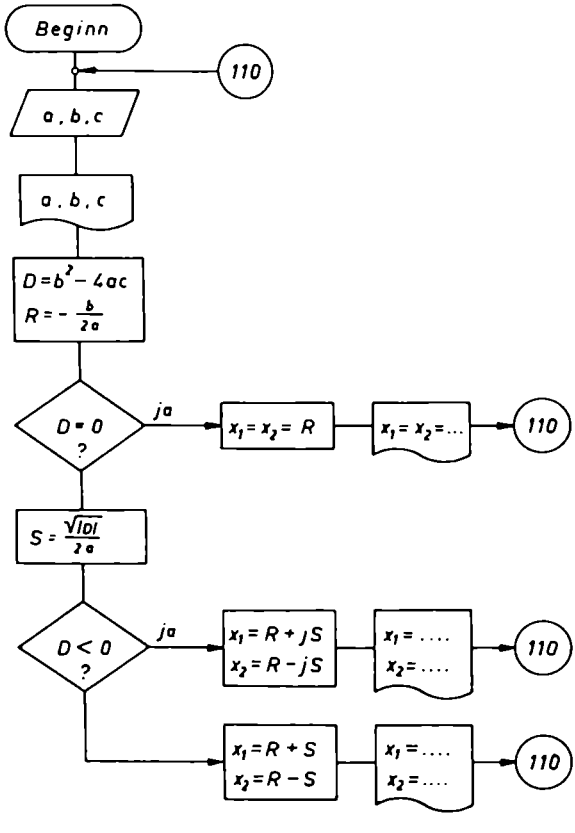
Quadratische Gleichung

Die Lösungszahlen (Wurzeln) der quadratischen Gleichung

$$ax^2+bx+c=0 \qquad (a\neq 0)$$

sind zu ermitteln.

Zunächst überlegen wir uns die möglichen Fälle und halten dann den Programmablauf graphisch fest.



Um die Richtigkeit des Programms überprüfen zu können, müssen die Testbeispiele alle möglichen Fälle enthalten.

Wir wählen die folgenden Gleichungen:

$x^2+x+1=0$	Wurzeln:	$x_1=-\frac{1}{2}+\frac{\sqrt{3}}{2}j; x_2=-\frac{1}{2}-\frac{\sqrt{3}}{2}j$
$x^2-4x+3=0$		$x_1=3 \qquad x_2=1$
$x^2-2x+1=0$		$x_1=x_2=1$

Programm:

```
100 PRINT " " ,"A*X↑2 + B*X + C = 0"
110 READ A,B,C
120 PRINT
130 PRINT "A =";A,"B =";B,"C =";C
140 D = B*B-4*A*C
150 R = -B/2/A
160 IF D = 0 THEN 210
170 S = SQR(ABS(D))/2/A
180 IF D < 0 THEN 230
190 PRINT "X1 =";R+S,"X2 =";R-S
200 GOTO 110
210 PRINT "X1 = X2 = ";R
220 GOTO 110
230 PRINT "X1 =";R;"+ J*";S,"X2 =";R;"- J*";S
240 GOTO 110
250 DATA 1,1,1, 1,-4,3, 1,-2,1
```

Ausgabe:

```
                A*X↑2 + B*X + C = 0

A = 1           B = 1           C = 1
X1 =-0.5 + J* 0.866025          X2 =-0.5 - J* 0.866025

A = 1           B =-4           C = 3
X1 = 3          X2 = 1

A = 1           B =-2           C = 1
X1 = X2 =  1
OUT OF DATA IN 110
```

Kubische Gleichung

Es sind die Wurzeln der Gleichung

$$x^3+ax^2+bx+c=0$$

mit Hilfe der Cardano-Formeln zu berechnen.

Die obige Gleichung wird durch die Substitution $x=y-\frac{a}{3}$ in die reduzierte Form[1]

$$y^3+3py+2q=0$$

übergeführt. Dabei gelten folgende Beziehungen:

$$3p=-\frac{a^2}{3}+b \qquad 2q=\frac{2a^3}{27}-\frac{ab}{3}+c$$

Die Diskriminante D der Gleichung $y^3+3py+2q=0$ lautet: $D=q^2+p^3$

Wir unterscheiden drei Fälle: $D>0$, $D=0$, $D<0$

Für $D>0$ lauten die Lösungen:

$$y_1=u+v \qquad x_1=y_1-\frac{a}{3}$$

$$y_2=-\frac{u+v}{2}+\mathrm{j}\frac{\sqrt{3}}{2}(u-v) \qquad x_2=y_2-\frac{a}{3}$$

$$y_3=-\frac{u+v}{2}-\mathrm{j}\frac{\sqrt{3}}{2}(u-v) \qquad x_3=y_3-\frac{a}{3}$$

mit $u=\sqrt[3]{-q+\sqrt{D}}$ und $v=\sqrt[3]{-q-\sqrt{D}}$

Für $D=0$ ist $u=v=\sqrt[3]{-q}$, folglich:

$$y_1=2\sqrt[3]{-q} \qquad x_1=y_1-\frac{a}{3}$$

$$y_2=y_3=-\sqrt[3]{-q} \qquad x_2=x_3=y_2-\frac{a}{3}$$

Im Fall $D<0$ besitzt die Gleichung drei reelle verschiedene Lösungen, nämlich

$$y_1=2\sqrt{-p}\cos\frac{\varphi}{3} \qquad x_1=y_1-\frac{a}{3}$$

$$y_2=-2\sqrt{-p}\cos\left(\frac{\varphi}{3}+60°\right) \qquad x_2=y_2-\frac{a}{3}$$

$$y_3=-2\sqrt{-p}\cos\left(\frac{\varphi}{3}-60°\right) \qquad x_3=y_3-\frac{a}{3}$$

wobei $\cos\varphi=\frac{-q}{\sqrt{-p^3}}$ und damit $\varphi=\arccos\left(\frac{-q}{\sqrt{-p^3}}\right)$ gilt.

[1] diese enthält kein quadratisches Glied!

Beim Programmieren müssen wir beachten:

a) $\sqrt[3]{z} = \operatorname{sgn}(z) \cdot \sqrt[3]{|z|}$

b) Wir müssen die Arkuskosinusfunktion durch die Arkustangensfunktion ausdrücken.

$$\arccos x = \begin{cases} \arctan \dfrac{\sqrt{1-x^2}}{x} & \text{für} \quad 10^{-8} < x \leqq 1 \\ \dfrac{\pi}{2} & \text{für} \quad |x| \leqq 10^{-8} \\ \pi + \arctan \dfrac{\sqrt{1-x^2}}{x} & \text{für} \quad -1 \leqq x < -10^{-8} \end{cases}$$

Die Testbeispiele müssen alle möglichen Fälle erfassen. Wir wählen daher folgende Gleichungen zur Überprüfung unseres Programms:

$x^3 - 19x^2 + 81x + 101 = 0$	$(x_1 = -1;\ x_2 = 10 + j;\ x_3 = 10 - j)$
$x^3 - 9x^2 + 24x - 16 = 0$	$(x_1 = 1;\ x_2 = x_3 = 4)$
$x^3 - 15x^2 + 75x - 125 = 0$	$(x_1 = x_2 = x_3 = 5)$
$x^3 - 13x^2 + 20x + 100 = 0$	$(x_1 = 10;\ x_2 = 5;\ x_3 = -2)$
$x^3 - 6x^2 = 0$	$(x_1 = 6;\ x_2 = x_3 = 0)$

Den Benützern des Time-Sharing-Systems der Firma Honeywell Bull steht außer Mark I-BASIC auch Mark III-BASIC zur Verfügung.

Diese umfangreichere und leistungsfähigere Sprache wird im Folgeband „BASIC-Praxis" ausführlich besprochen.

Hier sei nur festgestellt, daß Mark-I-Programme auch in Mark III ausführbar sind, wenn die END-Anweisung als Anweisung mit größter Zeilennummer explizit angeschrieben wird[1].

Wir stellen per Telefon die Verbindung mit der in den USA befindlichen Time-Sharing-Zentrale her.

Nach mehrmaligem Drücken der H-Taste meldet sich der Rechner. Wir geben unsere Nummer, unser Kennwort, das gewünschte System (BASIC), NEW und den von uns gewählten Programmnamen ein.

Nach der Rückmeldung READY bewirkt TAP die Umschaltung auf Lochstreifeneingabe. Nach der neuerlichen Meldung READY schalten wir den Lochstreifenleser ein.

Nach dem Einlesen des Programms bleibt der Lochstreifenleser stehen. Durch den Systembefehl RUN veranlassen wir dann die Ausführung des Programms.

```
HHHHHH
U#=HOS30000,
PASSWORD
SYSTEM- BAS
NEW OR OLD-
NEW KUB
```

[1] Außerdem muß RND(0) durch RND ersetzt werden und RND(−1) auf RAN (eigene Zeile!) geändert werden.
Es ist zu berücksichtigen, daß Rechner-Unendlich und Rechnerauflösungsvermögen in Mark III andere Werte als in Mark I haben.

Programm:

```
100 PRINT "      X↑3 + A*X↑2 + B*X + C = 0"
110 DEFFNZ(X) = SGN(X)*ABS(X)↑(1/3)
120 L = 1.04719755
130 READ A,B,C
140 PRINT
150 PRINT
160 PRINT "A = ";A,"B = ";B,"C = ";C
170 A3 = A/3
180 P = (B-A*A3)/3
190 Q = (2*A3↑3-A3*B+C)/2
200 D = Q↑2+P↑3
210 IF ABS(D) < 1E-10 THEN 340
220 IF D < 0 THEN 390
230'          D > 0
240 U = FNZ(-Q+SQR(D))
250 V = FNZ(-Q-SQR(D))
260 Y1 = U+V
270 R = -Y1/2
280 I = .8660254*(U-V)
290 PRINT "X1 = ";Y1-A3
300 PRINT "X2 = ";R-A3;"+J*";I
310 PRINT "X3 = ";R-A3;"-J*";I
320 GOTO 130
330'          D = 0
340 U = FNZ(-Q)
350 X1 = 2*U-A3
360 X2 = X3 =-U-A3
370 GOTO 470
380'          D < 0
390 K = -Q/((-P)↑1.5)
400 IF ABS(K) < 1E-10 THEN 530
410 B = ATN(SQR(1-K↑2)/K)/3
420 IF K < 0 THEN 510
430 X0 = 2*SQR(-P)
440 X1 = X0*COS(B)-A3
450 X2 = -X0*COS(B+L)-A3
460 X3 = -X0*COS(B-L)-A3
470 PRINT "X1 = ";X1
480 PRINT "X2 = ";X2
490 PRINT "X3 = ";X3
500 GOTO 130
510 B = L+B
520 GOTO 430
530 B = L/2
540 GOTO 430
550 DATA -19,81,101,  -9,24,-16,  -15,75,-125
560 DATA -13,20,100,  -6,0,0
570 END
```

Beachte:

Der Mark-III-Anweisung
230' D>0
entspricht in Mark I:
230 REM D>0

Ausgabe:

```
X↑3 + A*X↑2 + B*X + C = 0


A = -19          B =  81          C =  101
X1 = -1.
X2 =  10 +J* 1.
X3 =  10 -J* 1.


A = -9           B =  24          C = -16
X1 =  1
X2 =  4
X3 =  4


A = -15          B =  75          C = -125
X1 =  5
X2 =  5
X3 =  5


A = -13          B =  20          C =  100
X1 =  10.
X2 =  5.
X3 = -2.


A = -6           B =  0           C =  0
X1 =  6
X2 =  0
X3 =  0
OUT OF DATA IN 130


USED      .27 UNITS
BYE
0001.23  CRU    0000.19  TCH    0003.96  KC
```

In der mit USED beginnenden Zeile gibt der Rechner die für die Durchführung unseres Programms benötigte Zeit bekannt (UNIT ≙ Einheit).

Durch den Systembefehl BYE wird die Ausgabe von verbrauchter Rechenzeit[1], Anschaltzeit[2], Gesamte Zahl der übertragenen Zeichen[3] und des Datums (Monat, Tag, Jahr) veranlaßt und die Datenstation vom Rechner getrennt.

[1] CRU („computer resource units") Hauptrechnereinheiten
[2] TCH („terminal connect hours") Anschaltstunden
[3] KC („kilocharacters") 1 KC = 1000 übertragene Zeichen

Reelle Nullstellen einer Funktion

Ein wichtiges Problem der numerischen Mathematik ist die Berechnung der Nullstellen einer stetigen Funktion f innerhalb eines vorgegebenen Intervalls [A, E].

Wir beschränken uns auf ein einfaches, schnell zum Erfolg führendes Verfahren, das

Suchen der Nullstellen mittels Intervallschachtelung.

Wir teilen das Intervall [A, E] in 64 Teilintervalle:

[A, A+I], [A+I, A+2I], ...[1]

Wir suchen dann jenes Teilintervall $[x_k, x_{k+1}]$, bei dem die Vorzeichen von $f(x_k)$ und $f(x_{k+1})$ verschieden sind. In diesem Teilintervall liegt mindestens eine Nullstelle.

Zur Ermittlung der in $[x_k, x_{k+1}]$ gelegenen Nullstellen teilen wir x_k, x_{k+1} wieder in 64 Teilintervalle.

Wir wiederholen dieses Verfahren, bis

1. $f(x_n)=0$ oder
2. $x_n=x_{n+1}$

gilt[2].

Nach dem Ausdrucken der Nullstelle x_n und des zugehörigen Funktionswerts $f(x_n)$ müssen wir noch im Restintervall $(x_n, E]$ weitersuchen.

Als linke Begrenzung wählen wir, falls $x \neq 0$ ist, einfachheitshalber $A=x_n+|x_n| \cdot 10^{-8}$.

Falls $x=0$ ist, wählen wir $A=10^{-8}$.

Wir wiederholen das Verfahren so lange, bis wir an der Intervallsgrenze E anlangen[3]. Dann wird die Rechnung abgebrochen.

Für den Test wählen wir: $f(x)=x^4-5x^2+4$ $\quad A=-15$ $\quad E=15$

Ausdrücklich sei darauf verwiesen, daß bei der im Programm angewendeten Methode Nullstellen mit gerader Vielfachheit (es gilt dann: $\operatorname{sign} f(x_0-\varepsilon)=\operatorname{sign} f(x_0+\varepsilon)$) überhaupt nicht entdeckt werden, außer man stößt bereits bei der ersten Intervallteilung durch Zufall auf sie.

[1] 64 ist willkürlich angenommen; $I=(E-A)/64$.

[2] Rechnergenauigkeit!

[3] Beachte, daß es infolge der beschränkten Genauigkeit des Rechners vorkommen kann, daß die obere Grenze E bei diesem Verfahren nicht erreicht wird.

Programm:

```
100 REM  R E E L L E  N U L S T E L L E N
110 REM  VON F(X) IM INTERVALL (A,E)
120 REM  F(X) IST IN ZEILE 200 DEFINIERT
140 PRINT "REELLE NULLSTELLEN:"
150 PRINT " X","  F(X)"
200 DEFFNF(X) = X↑4 - 5*X↑2 + 4
210 READ A,E
220 E0 = E
230 I = (E-A)/64
240 IF A+I = A THEN 500                    ...1
250 Y0 = FNF(A)
255 REM ***** INTERVALL (X(K),X(K+1)) *****
260 FOR X = A TO E STEP I
270 Y = FNF(X)
280 IF ABS(Y) < 1E-8 THEN 500
290 IF SGN(Y) <> SGN(Y0) THEN 400
300 Y0 = Y
310 NEXT X
320 PRINT "KEINE WEITEREN R. NULLSTELLEN"
330 STOP
395 REM   SGN(F(X(K))) <> SGN(F(X(K+1)))
400 A = X-I
410 E = X
420 GOTO 230
495 REM ***** X IST NULLSTELLE     *********
500 PRINT X,Y
510 E = E0
520 IF X = 0 THEN 550
530 A = X + ABS(X)*1E-8
540 GOTO 230
550 A = 1E-8                               ...2
560 GOTO 230
570 DATA -15, 15
```

Ausgabe:

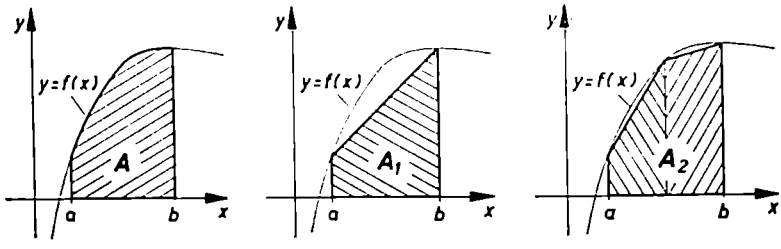
```
REELLE NULLSTELLEN:
 X                F(X)
-2.              -1.43051E-6
-1                0
 1                0
 2.               4.29154E-6
KEINE WEITEREN R. NULLSTELLEN
```

---

[1] Diese Abfrage basiert auf der beschränkten Genauigkeit des Rechners.
[2] Nullstellen mit $|x| < 10^{-8}$ werden hier nicht berücksichtigt!

Numerische Integration

Den Inhalt A der Fläche unter der Kurve mit $y=f(x)$ von $x=a$ bis $x=b$, also $\int_a^b f(x)\,dx$, kann man z. B. mit Hilfe der *Sehnen-Trapezregel* berechnen.



Es gilt:

$$A_1=\frac{f(a)+f(b)}{2}\cdot h \qquad \text{mit } h=b-a$$

$$A_2=\frac{f(a)+2f\left(a+\frac{h}{2}\right)+f(b)}{2}\cdot\frac{h}{2}$$

$$A_3=\frac{f(a)+2\left[f\left(a+\frac{h}{4}\right)+f\left(a+\frac{2h}{4}\right)+f\left(a+\frac{3h}{4}\right)\right]+f(b)}{2}\cdot\frac{h}{4} \text{ usw.}$$

Die Rechnung soll dann beendet werden, wenn sich die beiden letzten A-Werte um weniger als $A\cdot 10^{-5}$ unterscheiden.

Aus wirtschaftlichen Gründen werden wir unser Programm so aufbauen, daß jeder Funktionswert nur einmal berechnet wird[1].

Wir formen daher unsere Formeln folgendermaßen um:

$$2A_1=[f(a)+f(b)]\cdot h=S_1\cdot h$$

$$2A_2=\left[S_1+2f\left(a+\frac{h}{2}\right)\right]\cdot\frac{h}{2}=[S_1+2T_1]\cdot\frac{h}{2}=S_2\cdot\frac{h}{2}$$

$$2A_3=\left\{S_2+2\left[f\left(a+\frac{h}{4}\right)+f\left(a+\frac{3h}{4}\right)\right]\right\}\cdot\frac{h}{4}=[S_2+2T_2]\cdot\frac{h}{4}=$$
$$=S_3\cdot\frac{h}{4}$$

.....

[1] *Beachte:* Sowohl in $A_1, A_3, A_5, \ldots$ kommen die Werte $f(a)$ und $f(b)$ vor; sowohl in A_2 als auch in A_3 kommt $f\left(a+\frac{h}{2}\right)$ vor. Usw.

Zum Testen des Programms wählen wir das Polynom $1+2x+3x^2+4x^3$ und die Grenzen $a=1$ und $b=3$. Der exakte Wert des Flächeninhalts beträgt in diesem Fall

$$\int_1^3 (1+2x+3x^2+4x^3)\,dx = [x+x^2+x^3+x^4]_1^3 = 120-4=116$$

Programm:

```
100 DEFFNY(X) = 1+2*X+3*X↑2+4*X↑3
110 READ A,B
120 S = FNY(A)+FNY(B)
130 F = S*(B-A)
140 K = 1
150 K = 2*K
160 Z = (B-A)/K
170 Q = F
180 T = 0
190 FOR I = 1 TO K/2
200 T = T+FNY(A+(2*I-1)*Z)
210 NEXT I
220 S = S+2*T
230 F = S*Z
240 IF ABS(F-Q) > ABS(F)*1E-5 THEN 150
250 PRINT "FLAECHENINHALT =";F/2
260 GOTO 110
300 DATA 1,3
```

Ausgabe:

```
FLAECHENINHALT = 116.

OUT OF DATA  IN 110
```

ÜBUNGEN

13.01 Was muß man am Programm ändern, damit man mit diesem den Inhalt eines Halbkreises mit $r=10$ cm mit Hilfe der Trapezregel berechnen kann?

13.02 Wie ist das Programm zu ändern, wenn man die Fläche unter der Sinuskurve von $x=0$ bis $x=\pi$ erhalten will[1]?

13.03 Der Inhalt des unter der Kurve mit $y=\sin^2 x$ liegenden Flächenstücks zwischen $x=0$ und $x=\pi$ ist zu ermitteln[2]. Welche Änderung muß am Programm durchgeführt werden?

---

[1] Es gilt: $A=\int_0^\pi \sin x\,dx = -\cos x\Big|_0^\pi = -\cos\pi+\cos 0 = 1+1=2$

[2] Es gilt: $A=\int_0^\pi \sin^2 x\,dx = \frac{\pi}{2}$

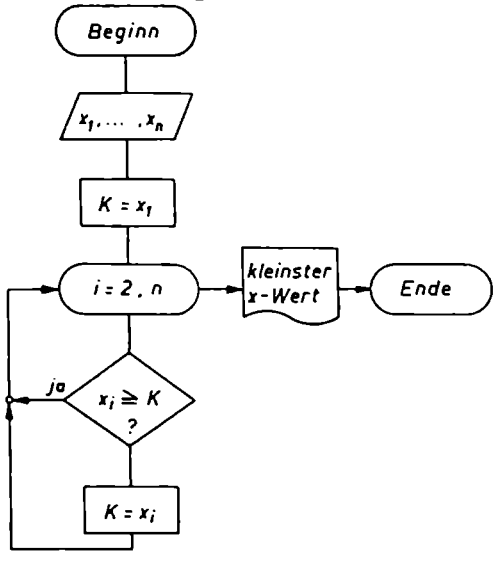
Sortierverfahren

In der Datenverarbeitung spielen Sortierverfahren eine große Rolle. Die Grundlage dieser Verfahren ist das Sortieren von Zahlen, das Ordnen einer Zahlenfolge der Größe nach.

BEISPIELE

1. Gegeben seien n Zahlen $x_1, x_2, \ldots, x_n$.
Es ist die kleinste dieser Zahlen zu suchen und auszudrucken.

Programmablaufplan:



Programm:

```
100 DIM X(100)
110 READ N
120 FOR J = 1 TO N
130 READ X(J)
140 NEXT J
150 K = X(1)                    Wir betrachten x1 als
160 FOR I = 2 TO N              vorläufig kleinste Zahl!
170 IF X(I) >= K THEN 190
180 K = X(I)
190 NEXT I
200 PRINT "KLEINSTE ZAHL:";K
210 DATA 6, 7.6, 0, 13,-3.5, 17.32, -8.4
```

Ausgabe:

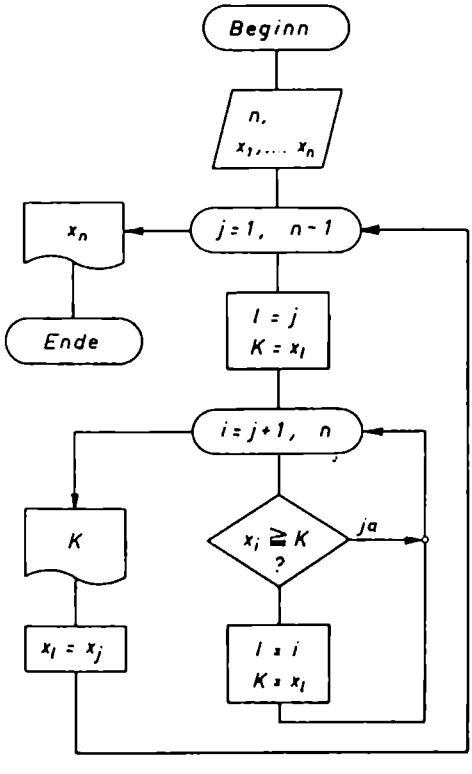
```
KLEINSTE ZAHL:-8.4
```

2. Wir haben im ersten Beispiel ein Programm erstellt, das die kleinste von n Zahlen $x_1, x_2, \ldots, x_n$ ermittelt. Wir wollen nun ein Sortierprogramm schreiben, das auf folgendem Prinzip beruht: Im ersten Durchgang wird die kleinste Zahl ermittelt und mit x_1 vertauscht. Sodann wird die kleinste Zahl von $x_2, \ldots, x_n$ gesucht und mit x_2 vertauscht usw.

Um den jeweiligen Zahlentausch $x_l \leftrightarrow K$ durchführen zu können, müssen wir den Index der jeweils kleinsten Zahl speichern; wir müssen also festhalten, an welcher Stelle sich die kleinste Zahl befunden hat.

Bezeichnungen:

x_i Zahlen, die zu sortieren sind
n Anzahl dieser Zahlen (Wir wählen $n \leqq 100$)
i laufender Index
K vorläufig kleinste Zahl
j Anfangsindex des jeweiligen Vergleichs
l Index der vorläufig kleinsten Zahl des jeweiligen Durchgangs, also jener Zahl, die als K notiert wurde.



Programm:

```
100 DIM X(100)
110 READ N
120 FOR I = 1 TO N
130 READ X(I)
140 NEXT I
150 FOR J = 1 TO N-1
160 L = J
170 K = X(L)
180 FOR I = J+1 TO N
190 IF X(I) >= K THEN 220
200 L = I
210 K = X(L)
220 NEXT I
230 PRINT K
240 X(L) = X(J)
250 NEXT J
260 PRINT X(N)
270 DATA 5
280 DATA 7, 2, 10, 2, -6
```

Ausgabe:

```
-6
 2
 2
 7
 10
```

3. Abschließend besprechen wir noch ein anderes leicht durchschaubares Sortierprinzip[1]; wir nennen es *Sortieren durch Dreieckstausch.*

Zunächst wird x_1 mit x_2 verglichen. Ist $x_1 < x_2$, so wird x_1 mit x_3 verglichen usw. Ist jedoch $x_2 < x_1$, so wird x_1 vom Speicher X(1) in den Hilfsspeicher H umgespeichert. Sodann wird x_2 nach X(1) gebracht und anschließend x_1 von H nach X(2). Bei jedem Austauschen zweier Zahlen wird eine Variable W auf 1 gesetzt. Man spricht von der *Weiche W* oder vom *Schalter W*, weil über den Inhalt von W Einfluß auf den Verlauf des Programms genommen wird.

Das oben beschriebene Verfahren wird so lange fortgesetzt, bis der bei der Weiche W am Anfang jeder Iteration gestellte Wert 0 erhalten bleibt[2].

[1] Die n Zahlen $x_1, x_2, \ldots, x_n$ sind nach steigender Größe zu ordnen.

[2] In diesem Fall wird während eines vollständigen Durchgangs kein Zahlenaustausch mehr vorgenommen, die Zahlen sind dann bereits nach aufsteigender Größe geordnet.

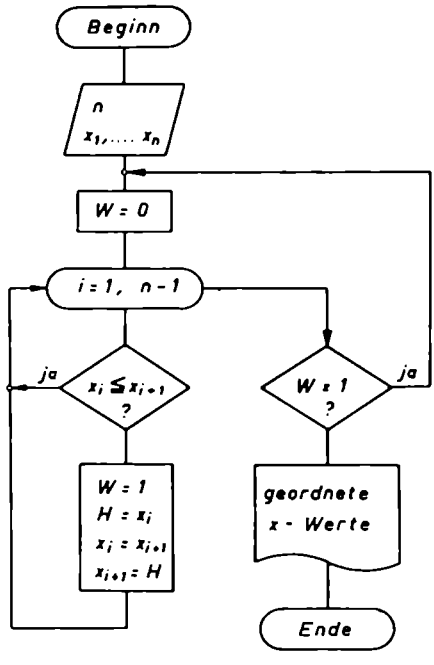
Zahlenbeispiel:

$n=5 \quad x_1=7 \quad x_2=2$
$x_3=10 \quad x_4=2 \quad x_5=0$

Leicht ist zu erkennen[1]:

Anfangs-zustand	nach 1.Durchg.	nach 2.Durchg.
7	2	2
2	7	2
10	2	0
2	0	7
0	10	10
$W=0$	$W=1$	$W=1$

nach 3.Durchg.	nach 4.Durchg.	nach 5.Durchg.
2	0	0
0	2	2
2	2	2
7	7	7
10	10	10
$W=1$	$W=1$	$W=0$



Programm:

```
100 DIM X(100)
110 READ N
120 FOR I = 1 TO N
130 READ X(I)
140 NEXT I
150 W = 0
160 FOR I = 1 TO N-1
170 IF X(I) <= X(I+1) THEN 220
180 W = 1
190 H = X(I)
200 X(I) = X(I+1)
210 X(I+1) = H
220 NEXT I
230 IF W = 1 THEN 150
240 FOR I = 1 TO N
250 PRINT X(I);
260 NEXT I
270 DATA 5
280 DATA 7, 2, 10, 2, 0
```

Ausgabe:

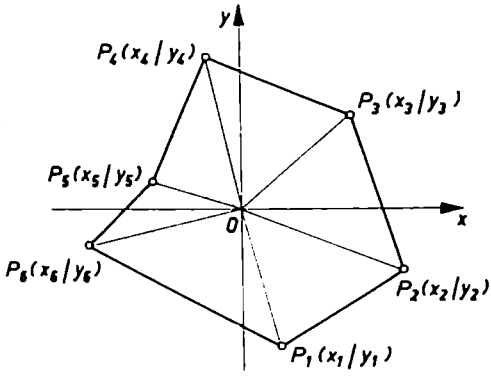
```
 0   2   2   7   10
```

[1] Verfolgen Sie den Vorgang genau anhand des Programmablaufplans!

Berechnung von Querschnittswerten

Sowohl für Maschinenbauer als auch für Bauingenieure ist die Berechnung von Flächeninhalt und Trägheitsmomenten von geradlinig begrenzten Querschnitten von grundlegender Bedeutung.

Wenn ein n-Eck durch die Koordinaten der Eckpunkte, also durch $(x_1 \mid y_1)$, $(x_2 \mid y_2)$, ..., $(x_n \mid y_n)$ gegeben ist, so gelten folgende Formeln:



Flächenformel von Gauß:

$$A = \frac{1}{2} \sum_{k=1}^{n} (x_k y_{k+1} - x_{k+1} y_k)$$

Koordinaten des Schwerpunkts:

$$x_s = \frac{M_y}{A} = \frac{\frac{1}{6} \sum_{k=1}^{n} (x_k y_{k+1} - x_{k+1} y_k)(x_k + x_{k+1})}{A}$$

$$y_s = \frac{M_x}{A} = \frac{\frac{1}{6} \sum_{k=1}^{n} (x_k y_{k+1} - x_{k+1} y_k)(y_k + y_{k+1})}{A}$$

Beachte:

$P_{n+1} = P_1$

$(x_{n+1} = x_1;\ y_{n+1} = y_1)$

Trägheitsmomente:

$$I_x = \int y^2 \mathrm{d}A = \frac{1}{12} \sum_{k=1}^{n} (x_k y_{k+1} - y_k x_{k+1})[(y_k + y_{k+1})^2 - y_k y_{k+1}]$$

$$I_y = \int x^2 \mathrm{d}A = \frac{1}{12} \sum_{k=1}^{n} (x_k y_{k+1} - y_k x_{k+1})[(x_k + x_{k+1})^2 - x_k x_{k+1}]$$

$$I_{xy} = \int xy \mathrm{d}A = \frac{1}{12} \sum_{k=1}^{n} \left\{ (x_k y_{k+1} - y_k x_{k+1}) \left[(x_k + x_{k+1})(y_k + y_{k+1}) - \right. \right.$$
$$\left. \left. - \frac{1}{2} (x_k y_{k+1} + y_k x_{k+1}) \right] \right\}$$

$$I_{s_x} = I_x - y_s^2 A \qquad I_{s_y} = I_y - x_s^2 A \qquad I_{s_x s_y} = I_{xy} - x_s y_s A$$

Hauptträgheitsmomente:

$$I_1 = \frac{I_{s_x} + I_{s_y}}{2} + \frac{I_{s_x} - I_{s_y}}{2} \cos 2\alpha_0 - I_{s_x s_y} \sin 2\alpha_0$$

$$I_2 = \frac{I_{s_x} + I_{s_y}}{2} - \frac{I_{s_x} - I_{s_y}}{2} \cos 2\alpha_0 + I_{s_x s_y} \sin 2\alpha_0$$

mit $\tan 2\alpha_0 = \dfrac{-2 I_{s_x s_y}}{I_{s_x} - I_{s_y}}$

Es gilt: $I_p = I_{s_x} + I_{s_y} = I_1 + I_2$

Programm:

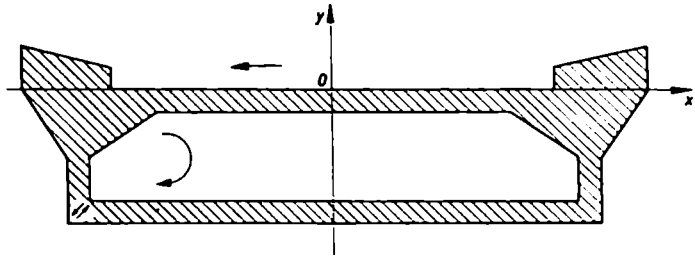
```
100 DIM X(50), Y(50)
110 READ N
120 PRINT
130 PRINT
140 PRINT "Q U E R S C H N I T T S W E R T E"
150 PRINT
160 PRINT "ECKPUNKTE:"
170 PRINT " NR","  X","  Y"
180 FOR K = 1 TO N
190   READ X(K),Y(K)
200   PRINT K,X(K),Y(K)
210 NEXT K
215
220 A = M1 = M2 = I1 = I2 = I3 = K = 0
230 R = X(N)
240 S = Y(N)
250 FOR K = 1 TO N
260   P = R
270   Q = S
280   R = X(K)
290   S = Y(K)
300   D = P*S-Q*R
310   A = A+D
320   M1 = M1+D*(Q+S)
330   M2 = M2+D*(P+R)
340   I1 = I1+D*(Q*Q+Q*S+S*S)
350   I2 = I2+D*(P*P+P*R+R*R)
360   I3 = I3+D*(P*Q+R*S+(P*S+Q*R)/2)
370 NEXT K
380 A = A/2
390 M1 = M1/6
400 M2 = M2/6
410 I1 = I1/12
420 I2 = I2/12
430 I3 = I3/12
440 X0 = M2/A
450 Y0 = M1/A
455
460 PRINT
470 PRINT "A, SX, SY", A,X0,Y0
480 PRINT
490 PRINT "STAT.MOMENTE",M1,M2
500 PRINT
510 PRINT "TRAEGHEITSMOMENTE"
520 PRINT "IX, IY, IXY",I1,I2,I3
530 I4 = I1-A*Y0^2
540 I5 = I2-A*X0^2
550 I6 = I3-A*X0*Y0
560 PRINT "ISX, ISY, ISXY",I4,I5,I6
570 Q1 = (I4+I5)/2
580 Q2 = (I5-I4)/2
590 IF ABS(I6) < ABS(Q1)*1E-6 THEN 640
600   IF ABS(Q2) > ABS(Q1)*1E-6 THEN 630
610     W = 1.57079633
620   GOTO 640
630   W = ATN(I6/Q2)
640 I7 = Q1-Q2*COS(W)-I6*SIN(W)
650 PRINT
660 PRINT "HAUPTTRAEGHEITSMOMENTE",I7,I4+I5-I7
670 PRINT
680 PRINT "DREHWINKEL",W*28.64788981"GRAD"
690 PRINT
700 PRINT "**********************************************************"
710 GOTO 110
```

Beachte:

M1 $\triangleq M_x$
M2 $\triangleq M_y$
I1 $\triangleq I_x$
I2 $\triangleq I_y$
I3 $\triangleq I_{xy}$
I4 $\triangleq I_{s_x}$
I5 $\triangleq I_{s_y}$
I6 $\triangleq I_{s_x s_y}$
I7 $\triangleq I_1$
I8 $\triangleq I_2$

... Abfrage: $I_{s_x s_y} = 0$?
... Abfrage: $I_{s_x} \neq I_{s_y}$?

```
720 DATA 20
730 DATA -10,0,-10,1,-14,2,-14,0,-12,-3,-12,-6
740 DATA -11,-5,-11,-3,-8,-1,8,-1,11,-3,11,-5
750 DATA -11,-5,-12,-6,12,-6,12,-3,14,0,14,2,10,1,10,0
```



Ausgabe:

```
Q U E R S C H N I T T S - E R T E

ECKPUNKTE:
 NR              X              Y
 1             -10              0
 2             -10              1
 3             -14              2
 4             -14              0
 5             -12             -3
 6             -12             -6
 7             -11             -5
 8             -11             -3
 9              -8             -1
 10              8             -1
 11             11             -3
 12             11             -5
 13            -11             -5
 14            -12             -6
 15             12             -6
 16             12             -3
 17             14              0
 18             14              2
 19             13              1
 20             10              0

A, SX, SY        80              0                -2.18333

STAT.MOMENTE   -174.667          0

TRAEGHEITSMOMENTE
IX, IY, IXY      855.667         6737.67          0
ISX, ISY, ISXY   474.311         6737.67          0

HAUPTTRAEGHEITSMOMENTE           474.311          6737.67

DREHWINKEL       0 GRAD

**************************************************
```

Lineares Gleichungssystem (Verfahren von Gauß-Jordan)

Gegeben sei ein lineares Gleichungssystem in n Variablen:

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= a_{1\,n+1}\\
a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= a_{2\,n+1}\\
&\ldots\ldots\\
a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= a_{n\,n+1}
\end{aligned}$$

a_{ij}:
$i = 1, \ldots, n$
$j = 1, \ldots, n+1$

Auf Seiten 106/107 wurde gezeigt, wie ein lineares Gleichungssystem mittels Berechnung der inversen Matrix gelöst werden kann. Es gibt aber BASIC-Compiler, bei denen die Matrizenoperationen nicht einprogrammiert sind; dann bietet sich zur Lösung eines linearen Gleichungssystems das nachfolgend beschriebene *Verfahren von Gauß-Jordan* an.

Wir eliminieren zunächst x_1 aus allen Gleichungen, außer der ersten.

Dazu wird die erste Gleichung mit

$\frac{a_{21}}{a_{11}}$ multipliziert und dann von der zweiten Gleichung subtrahiert,

$\frac{a_{31}}{a_{11}}$ multipliziert und dann von der dritten Gleichung subtrahiert,

.....

$\frac{a_{n1}}{a_{11}}$ multipliziert und dann von der n-ten Gleichung subtrahiert.

Auf diese Weise erhält man das System:

$$\begin{aligned}
a_{11}'x_1 + a_{12}'x_2 + a_{13}'x_3 + \ldots + a_{1n}'x_n &= a_{1\,n+1}'\\
a_{22}'x_2 + a_{23}'x_3 + \ldots + a_{2n}'x_n &= a_{2\,n+1}'\\
&\ldots\ldots\\
a_{n2}'x_2 + a_{n3}'x_3 + \ldots + a_{nn}'x_n &= a_{n\,n+1}'
\end{aligned}$$

$$a_{ij}' = a_{ij} - \frac{a_{i1}}{a_{11}} \cdot a_{1j}$$
$i = 2, \ldots, n$
$j = 2, \ldots, n+1$

Beachte: 1. Die hier und im folgenden rechts außen angegebenen Reduktionsformeln werden nur für jene Koeffizienten angeschrieben, die man im Verlauf der weiteren Rechnung wieder benötigt.

2. Die Speicherplätze A(I,J) werden jetzt mit den neuen Koeffizienten a_{ij}' belegt.

Nun eliminiert man x_2 aus allen Gleichungen, außer der zweiten.

Die zweite Gleichung des obigen Systems wird mit

$\frac{a_{12}}{a_{22}'}$ multipliziert und dann von der ersten Gleichung subtrahiert,

$\frac{a_{32}'}{a_{22}'}$ multipliziert und dann von der dritten Gleichung subtrahiert,

$\frac{a_{n2}'}{a_{22}'}$ multipliziert und dann von der n-ten Gleichung subtrahiert.

Man erhält so das System:

$$\begin{aligned}
a_{11}''x_1 \quad &+ a_{13}''x_3 + \ldots + a_{1n}''x_n = a_{1\,n+1}'' \qquad a_{ij}'' = a_{ij}' - \frac{a_{i2}'}{a_{22}'} \cdot a_{2j}' \\
a_{22}''x_2 + a_{23}''x_3 &+ \ldots + a_{2n}''x_n = a_{2\,n+1}'' \\
a_{33}''x_3 &+ \ldots + a_{3n}''x_n = a_{3\,n+1}'' \qquad i = 1, 3, 4, \ldots, n \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad j = 3, \ldots, n+1 \\
a_{n3}''x_3 &+ \ldots + a_{nn}''x_n = a_{n\,n+1}''
\end{aligned}$$

Beachte: Die Speicherplätze A(I,J) werden jetzt mit den neuen Koeffizienten a_{ij}'' belegt.

Eliminiert man x_k aus allen Gleichungen, außer der k-ten, so erhält man das System:

$$\begin{aligned}
a_{11}^{(k)}x_1 \quad &+ a_{1,k-1}^{(k)}x_{k-1} + \ldots + a_{1n}^{(k)}x_n = a_{1,n+1}^{(k)} \qquad a_{ij}^{(k)} = a_{ij}^{(k-1)} - \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} \cdot a_{kj}^{(k-1)} \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
a_{kk}^{(k)}x_k + a_{k,k+1}^{(k)}x_{k+1} &+ \ldots + a_{kn}^{(k)}x_n = a_{k,n+1}^{(k)} \qquad k = 1, \ldots, n \\
a_{k+1,k-1}^{(k)}x_{k-1} + \ldots &+ a_{k+1,n}^{(k)}x_n = a_{k+1,n+1}^{(k)} \qquad i = 1, \ldots, n;\ i \neq k \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad j = k+1, \ldots, n+1 \\
a_{n,k-1}^{(k)}x_{k-1} &+ \ldots + a_{nn}^{(k)}x_n = a_{n,n+1}^{(k)}
\end{aligned}$$

Beachte: Die Speicherplätze A(I,J) werden jetzt mit den neuen Koeffizienten $a_{ij}^{(k)}$ belegt.

Eliminiert man x_n aus allen Gleichungen, außer der n-ten, so erhält man das System:

$$\begin{aligned}
a_{11}^{(n)}x_1 \qquad\qquad &= a_{1,n+1}^{(n)} \qquad a_{ij}^{(n)} = a_{ij}^{(n-1)} - \frac{a_{in}^{(n-1)}}{a_{nn}^{(n-1)}} \cdot a_{nj}^{(n-1)} \\
a_{22}^{(n)}x_2 \qquad\qquad &= a_{2,n+1}^{(n)} \\
\ldots\ldots \qquad\qquad &\qquad i = 1, \ldots, n-1 \\
\ldots\ldots \qquad\qquad & \\
a_{nn}^{(n)}x_n &= a_{n,n+1}^{(n)} \qquad j = n+1
\end{aligned}$$

Beachte: Die Speicherplätze A(I,J) werden jetzt mit den neuen Koeffizienten $a_{ij}^{(n)}$ belegt.

Die Berechnung der x_i ist nun leicht:

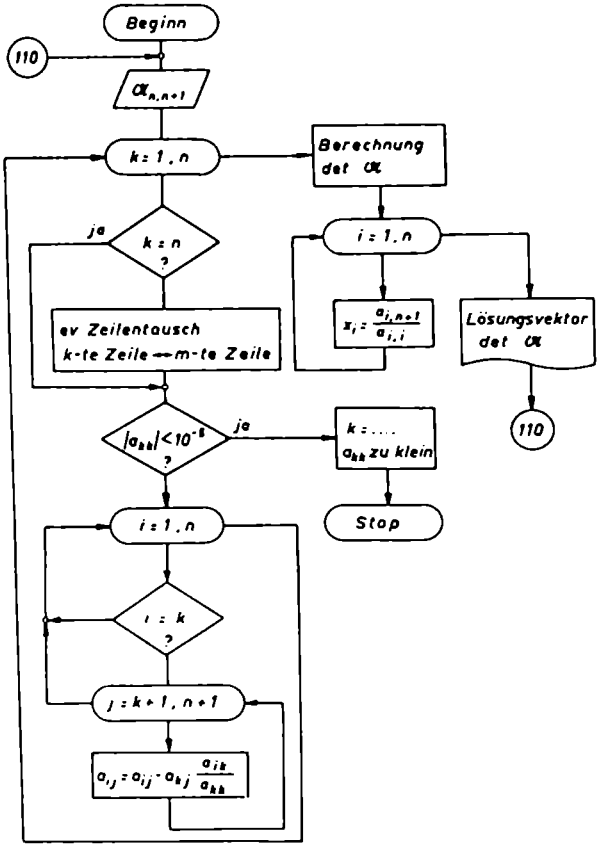
$$x_i = \frac{a_{i,n+1}^{(n)}}{a_{ii}^{(n)}} \quad \text{für} \quad i = 1, \ldots, n$$

Beachte: Das Produkt $a_{11}^{(n)} \cdot a_{22}^{(n)} \cdot \ldots \cdot a_{nn}^{(n)}$ ist gleich dem Wert der Determinante der Koeffizientenmatrix[1].

---

[1] Es gelten die Sätze:
Der Wert einer Determinante bleibt unverändert, wenn man zu den Elementen einer Zeile ein beliebiges Vielfaches einer anderen Zeile addiert.
Der Wert der Determinante einer Dreiecksmatrix ist gleich dem Produkt der Elemente ihrer Hauptdiagonale.

Wir zeichnen nun einen Programmablaufplan:



Programm:

```
100 DIM A(10,11)
110 READ N
120 PRINT "KOEFFIZIENTENMATRIX"
130 FOR I = 1 TO N
140 FOR J = 1 TO N+1
150 READ A(I,J)
160 PRINT A(I,J),
170 NEXT J
180 PRINT
190 NEXT I
200 PRINT
210 PRINT
220 FOR K = 1 TO N
230 IF K = N THEN 380
240: ***** TAUSCH: K.-ZEILE <- > M.-ZEILE *****
250 G = ABS(A(K,K))
260 M = K
270 FOR I = K+1 TO N
280 IF ABS(A(I,K)) <= G THEN 310
290 G = ABS(A(I,K))
300 M = I
310 NEXT I
320 FOR J = K TO N+1
330 Z = A(K,J)
340 A(K,J) = A(M,J)
350 A(M,J) = Z
360 NEXT J
```

```
370: ***  G.J.-ALGORITHMUS            **********
380 IF ABS(A(K,K)) < 1E-5 THEN 610
390 FOR I = 1 TO N
400 IF I = K THEN 450
410 F = A(I,K)/A(K,K)
420 FOR J = K+1 TO N+1
430 A(I,J) = A(I,J)-A(K,J)*F
440 NEXT J
450 NEXT I
460 NEXT K
470: ***  BERECHNUNG VON DET(A)       **********
480 D = 1
490 FOR I = 1 TO N
500 D = D*A(I,I)
510 NEXT I
520: ***  BERECHNUNG DER X(I)         **********
530 PRINT "LOESUNGSVEKTOR"," ","DET(A) ="D
540 FOR I = 1 TO N
550 PRINT A(I,N+1)/A(I,I)
560 NEXT I
570 PRINT
580 PRINT "****************************************"
590 PRINT
600 GOTO 110
610 PRINT "K ="K,"A(K,K) ZU KLEIN"
620 GOTO 570
630 DATA 2, 1,5,6, -2,4,-2
640 DATA 2, 3,4,5, 6,8,-3
650 DATA 3, 1,1,1,27, 2,-1,-1,0, 3,-2,-.25,5
```

Ausgabe:

```
KOEFFIZIENTENMATRIX
 1                5                6
-2                4               -2


LOESUNGSVEKTOR                    DET(A) =-14
 3.
 1

****************************************

KOEFFIZIENTENMATRIX
 3                4                5
 6                8               -3


K = 2           A(K,K) ZU KLEIN

****************************************

KOEFFIZIENTENMATRIX
 1                1                1               27
 2               -1               -1               0
 3               -2               -.25             5


LOESUNGSVEKTOR                    DET(A) =-5.25
 9.
 10.
 5.

****************************************


OUT OF DATA  IN 110
```

14. SYSTEMBEFEHLE

Systembefehle dienen zur Steuerung des Systems.

Wir beschränken uns hier auf die Angabe der wichtigsten Systembefehle des Betriebssystems Mark I/7 von General Electric.

Ein Systembefehl muß ohne Zeilennummer am Anfang einer Zeile stehen.

Auf den Befehl muß ein Wagenrücklauf (mittels RETURN-Taste!) folgen.

Die bei der Eingabe unbedingt erforderlichen Buchstaben sind hier Großbuchstaben, der Rest Kleinbuchstaben.

BYE	bewirkt die Trennung des Terminals vom Betriebssystem. Die Anschlußzeit und die verbrauchte Rechenzeit werden bekanntgegeben[1].
CATalog	bewirkt das Drucken einer Liste mit den Namen aller Programme, die unter der Teilnehmernummer gespeichert sind[2].
KEY	schaltet vom Lochstreifenbetrieb auf die Eingabe über die Terminaltastatur um[3].
LISt	bewirkt das Drucken einer Liste des im Arbeitsspeicher befindlichen Programms[4]. LIS 19Ø Liste des im Arbeitsspeicher befindlichen Programms ab Zeile 190.
LISTNH	bewirkt LIST ohne Kopfzeile. NH ist eine Abkürzung für no heading.
NEW Programmname	löscht das im Arbeitsspeicher befindliche Programm und speichert den neuen Namen. Der Arbeitsspeicher steht für das neue Programm zur Verfügung.
OLD Programmname	löscht das im Arbeitsspeicher befindliche Programm und holt das gewünschte Programm aus dem Bibliotheksspeicher (wo es weiterhin gespeichert bleibt!) in den Arbeitsspeicher.
RENname Programmname	bewirkt die Umbenennung des im Arbeitsspeicher befindlichen Programms[5]. (Das Programm wird nicht verändert!)
RUN	bewirkt die Ausführung des Programms[6]. Das im Arbeitsspeicher befindliche Programm wird nach Zeilennummern geordnet, mit Hilfe des BASIC-Compilers in die Maschinensprache übersetzt und anschließend durchgeführt.

---

[1] bye ≙ leb wohl
[2] catalog ≙ Katalog
[3] key ≙ Taste
[4] list ≙ Liste, auflisten
[5] rename ≙ umbenennen
[6] run ≙ laufen, Lauf

REPlace	bewirkt die Speicherung des im Arbeitsspeicher befindlichen Programms im Bibliotheksspeicher[1]. Dort wird ein unter gleichem Namen vorhandenes Programm gelöscht! Der Inhalt des Arbeitsspeichers bleibt unverändert.
SAVe	wirkt wie REP, wird jedoch nicht ausgeführt, wenn sich ein Programm gleichen Namens im Bibliotheksspeicher befindet. In diesem Fall erfolgt Meldung durch das Betriebssystem!
SCRatch	löscht das im Arbeitsspeicher befindliche Programm[2]. Der Programmname bleibt erhalten.
SYStem	ermöglicht den Wechsel zu einer anderen Programmiersprache[3].
TAPe	schaltet von der Eingabe über die Terminaltastatur auf Lochstreifenbetrieb um.
UNSave	löscht das Programm im Bibliotheksspeicher[4]. Der Inhalt des Arbeitsspeichers bleibt unverändert. UNS GEO, GLEI, SORT löscht die im Bibliotheksspeicher befindlichen Programme GEO, GLEI und SORT. (In Mark III heißt der entsprechende Befehl PUR GEO: GLEI; SORT)

15. EDIT-BEFEHLE

Den Benützern des Betriebssystems Mark I/7 (bzw. Mark III) steht für die Aufbereitung von Texten und Programmen die sehr wirkungsvolle Programmiersprache EDIT zur Verfügung.
Hier wird in Kurzform die Wirkungsweise der wichtigsten EDIT-Befehle aufgezeigt[5].

Beachte: EDIT-Befehle werden mit dem Befehlswort EDIT aufgerufen.

Bei allen EDIT-Befehlen werden Zeilennummern, zu denen in GOTO-, GOSUB- und IF-Anweisungen gesprungen werden soll, sinngemäß geändert.

EDIT DELETE Löschen von Zeilen[6]

EDI DEL 15Ø, 28Ø—31Ø, 37Ø

bewirkt im Arbeitsspeicher das Löschen der Zeilen 15Ø, 37Ø und aller Zeilen von 28Ø bis einschließlich 31Ø.

[1] replace △ ersetzen
[2] scratch △ ausstreichen, löschen, absagen
[3] system △ System, Programmiersprache
[4] unsave △ (im Bibliotheksspeicher) löschen
[5] edit △ aufbereiten
[6] delete △ streichen

EDIT DUPLICATE Duplizieren von Zeilen[1]

EDI DUP N1—N2, N3, N4

bewirkt, daß die in den Zeilen N1—N2 stehenden Anweisungen nach den Zeilen N3 und N4 im Arbeitsspeicher eingefügt werden.

Anschließend wird EDIT RES ausgeführt.

EDIT EXTRACT Löschen von Zeilen[2]

EDI EXT 15Ø, 28Ø—31Ø, 37Ø

bewirkt, daß alle Zeilen des Programms, außer den genannten, im Arbeitsspeicher gelöscht werden.

EDIT LIST Schreiben von Zeilen

EDI LIS 15Ø, 28Ø—31Ø, 37Ø—35Ø

bewirkt das Schreiben der Zeile 15Ø, aller Zeilen zwischen 28Ø bis einschließlich 31Ø und aller Zeilen zwischen 37Ø bis einschließlich 35Ø.

EDIT MERGE Verschmelzung von Programmen[3]

EDI MER DAT, TEIL 1, N1, TEIL 2, N2, ...

bewirkt, daß die Datei DAT in den Arbeitsspeicher gebracht wird und die Datei TEIL 1 nach der Zeile N1, die Datei TEIL 2 nach der Zeile N2, ... hinzugefügt wird.

Anschließend wird EDIT RES ausgeführt.

Beachte: Alle Daten müssen im Bibliotheksspeicher oder in der Systembibliothek gespeichert sein.

EDI MER DAT1, DAT2, DAT3

bewirkt die Verschmelzung dieser 3 Dateien.

Da keine Zeilennummern angegeben sind, wird DAT2 nach der letzten Zeile von DAT1 und DAT3 nach der letzten Zeile von DAT2 angefügt.

Das Gesamtprogramm steht unter dem Namen DAT3 im Arbeitsspeicher (neugeordnet ab Zeile 1ØØ mit der Schrittweite 1Ø) zur Verfügung.

EDIT MOVE Umstellung eines Programms[4]

EDI MOV N1—N2, N3

bewirkt, daß die Zeilen N1—N2 im Arbeitsspeicher nach der Zeile N3 eingefügt werden.

EDIT RESEQUENCE Neuordnung eines Programms[5]

EDI RES N1, N2—N3, N4

bewirkt, daß die Zeilen N2—N3 im Arbeitsspeicher neu mit der Anfangsnummer N1 und der Schrittweite N4 geordnet werden.

EDI RES

bewirkt Neuordnung ab 1ØØ mit Schrittweite 1Ø.

[1] duplicate ≙ duplizieren, vervielfältigen
[2] extract ≙ entnehmen
[3] merge ≙ verschmelzen, einverleiben
[4] move ≙ bewegen;
[5] resequence ≙ umnumerieren

LÖSUNGEN ZU DEN ÜBUNGEN

1.01 Programm:

```
10 PRINT 14*0.5*0.782
```

Ausgabe:

```
5.474
```

1.02 Programm:

```
10 PRINT 5*3-6*7+3/5, 7*8-16*7+8/32
```

Ausgabe:

```
-26.4         -55.75
```

1.03 a) 48 b) 50 −2

c) 2 6 6 9 27

d) 2 6 6
9 27

1.04 Programm:

```
10 PRINT "TESTPROGRAMM"
12 READ I,J
14 PRINT I,J,J/I
16 GOTO 12
18 DATA 8,2,0,7,3,9
```

Ausgabe:

```
TESTPROGRAMM
 8                2                .25
 0                7
DIVISION BY ZERO  IN 14
                                   5.78960E+76
 3                9                3

OUT OF DATA  IN 12
```

1.05 Am Blattschreiber des Terminals sind keine hochgestellten Ziffern und kein Wurzelzeichen vorhanden.

1.06 Programm:

```
10 PRINT "UEBUNG 1.06"
20 READ X
30 PRINT X, 1/X, SQR(X)
40 GOTO 20
50 DATA 1, -3, 0, 5
```

Ausgabe:

```
UEBUNG 1.06
 1                1                1
-3               -.333333
SQUARE ROOT OF NEGATIVE NUMBER  IN 30
                                 1.73205
 0
DIVISION BY ZERO  IN 30
                  5.78960E+76    0
 5                .2             2.23607

OUT OF DATA  IN 20
```

1.07 Programm:

```
10 PRINT "NEUGRAD","ALTGRAD"
20 READ N
30 PRINT N, .9*N
40 GOTO 20
50 DATA 100, 41.2778, 19.25
```

Ausgabe:

```
NEUGRAD          ALTGRAD
 100              90.
 41.2778          37.15
 19.25            17.325

OUT OF DATA  IN 20
```

1.08 Fortsetzung des Programms mit der in der Zeile 40 stehenden Anweisung.

2.01 a) Ganzzahl b) Gleitkommazahl
c) Fixkommazahl d) Ganzzahl
e) Fixkommazahl f) Gleitkommazahl

2.02 a) 3 8 b) 4 3
c) 5 1 d) 4 6
e) 3 7 f) 1 3

2.03 a) In BASIC muß statt des Kommas der Punkt verwendet werden. Wir haben vereinbart, daß die Null durchgestrichen wird.
c) $-.412.10^2$ ist in BASIC nicht erlaubt; es muß −Ø.412E2 heißen!
d) 5/3 ist in BASIC keine Zahlendarstellung, sondern eine Termschreibweise.

2.04 b) 5A Sollte A5 heißen!
d) und e) Nur ein Buchstabe darf als Kennzeichnung einer Variablen verwendet werden.

2.05 a) B/C b) $D \uparrow 2$ c) E/F

3.01 U−V+W

3.02 X−Y*Z

3.03 A*X+B*Y−C

3.04 R/S−T/U bzw. (R/S)−(T/U)

3.05 (X−3*Y)/Z

3.06 (D−4*E*F)/(2*G*H)−A↑2 bzw.
(D−4*E*F)/2/G/H−A↑2

3.07 (A−B)↑3

3.08 A−B↑3

3.09 (A−B)↑(2/3)

3.10 (A−2*B)*(2*A+3*B)

3.11 A↑(R−S)−B↑(2*R+3*S)

3.12 A↑(R−S)*B↑(2*R+S)

3.13 (A/(2*B))↑(−4)

3.14 (A/(−B))↑(−2)

3.15 A∗SIN(W2)/SIN(W1), wobei W1 das Bogenmaß von α und W2 das Bogenmaß von β ist[1].

3.16 SIN(W1)↑2 oder (SIN(W1))↑2

3.17 2∗SIN((W1+W2)/2)∗COS((W1−W2)/2) ...[1]

4.01—4.08

```
401 X = 3*A-2*B+C
402 Y = 5*A-D/C
403 Z = SQR(A↑2+B↑2+C↑2+D↑2)
404 P1 = (A*X+B)/(C*X+D)
405 P2 = (A*X↑2+B*X+C)/(B*X↑2-A*X+D)
406 S3 = 1+Q+Q↑2+Q↑3
407 C = 3.14159/180*W
408 X1 = (-B+SQR(B↑2-4*A*C))/(2*A)
```

Gespeicherter Wert:

```
2
3.66667
5.47723
.4 ...[2]
1.1
−20
.523598
−.309781
```

5.01 LOG

5.02 SQR, LOG

5.03 SQR(u)

u ... Rechner-Unendlich

a ... Rechner-Auflösungsvermögen

5.04 $a \leqq x \leqq u$ $\quad \ln a \leqq y \leqq \ln u$

5.05 $\ln a \leqq x \leqq \ln u$ $\quad a \leqq y \leqq u$

5.06 a) 7 b) 6 c) e d) 1,4 e) $\frac{\pi}{4}$

5.07 SIN((12+17/60+36/3600)/57.2957795)

5.08 .434294482∗LOG(132) oder LOG(132)/LOG(10)

5.09 ATN(3.2)∗57.2957795

5.10 ATN(3.2)∗200/3.14159265

5.11 INT(65∗RND(0)+13)

5.12 15.1∗RND(0)−9.8

5.13 INT(100∗X+.5)/100

5.14 SGN(X)∗INT(100∗ABS(X)+.5)/100

[1] Es können selbstverständlich auch andere Variablen verwendet werden.

[2] *Beachte:* Durch 401 X=3∗A−2∗B+C wurde X mit 2 belegt!

5.15 DEFFNS(X)=(EXP(X)−EXP(−X))/2

5.16 DEFFNC(X)=(EXP(X)+EXP(−X))/2

5.17 DEFFNT(X)=1/TAN(X)

5.18 DEFFNA(X)=SIN(X+3.14159265E−2)

5.19 DEFFNB(X)=LOG(X+SQR(X ↑ 2+1))

5.20 DEFFND(X)=LOG(X+SQR(X ↑ 2−1))

5.21 DEFFNE(X)=.5*LOG((1+X)/(1−X))

8.01 a) IF X < > 0 THEN 30

b) IF A < = B THEN 20

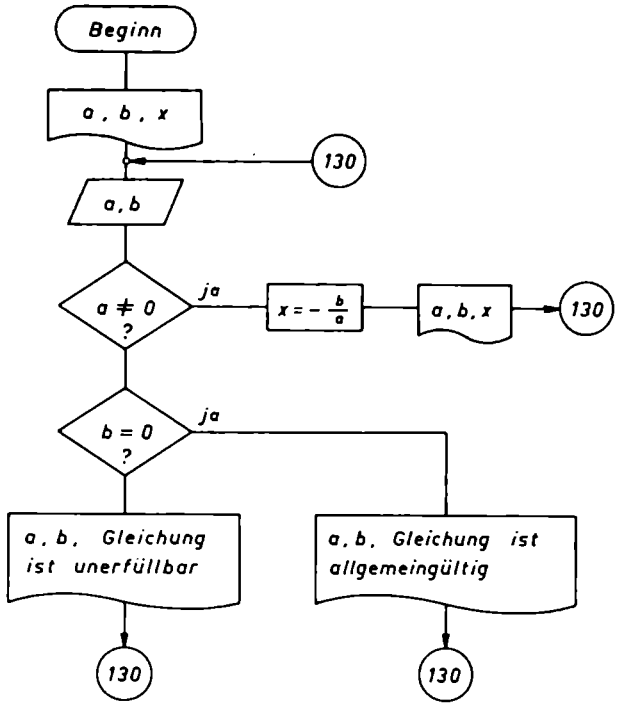
c) IF 2*X = SQR(Y) THEN 60

d)
```
30 IF A ↑ 2−4*A*B=0 THEN 90
40 IF A ↑ 2 > 4*A*B THEN 150
50 .....
```

e)
```
20 IF M < N THEN 60
25 IF M > N THEN 40
30 .....
```

8.02
```
150 IF X > 0 THEN 190
160 IF X < 0 THEN 210
170 Y = −2
180 GOTO 220
190 Y=3*X
200 GOTO 220
210 Y=2*X
220 Z=X+Y
```

8.03



Programm:

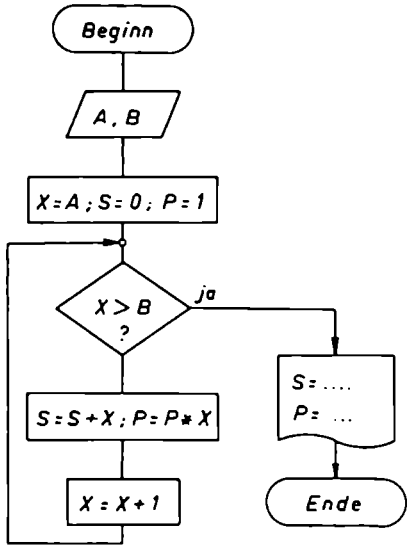
```
120 PRINT " A"," B"," X"
130 READ A,B
140 IF A = 0 THEN 180
150 X = -B/A
160 PRINT A,B,X
170 GOTO 130
180 IF B = 0 THEN 210
190 PRINT A,B,"GLEICHUNG IST UNERFUELLBAR"
200 GOTO 130
210 PRINT A,B,"GLEICHUNG IST ALLGEMEINGUELTIG"
220 GOTO 130
230 DATA 2, -6,  0, 0,  0, 5,  6, 0
```

Ausgabe:

```
 A              B              X
 2             -6              3
 0              0             GLEICHUNG IST ALLGEMEINGUELTIG
 0              5             GLEICHUNG IST UNERFUELLBAR
 6              0              0

OUT OF DATA  IN 130
```

8.04 Programmablaufplan:



Programm:

```
100 READ A,B
110 X = A
120 P = 1
130 IF X > B THEN 180
140 S = S+X
150 P = P*X
160 X = X+1
170 GOTO 130
180 PRINT "S = ";S,"P = ";P
190 DATA 1, 20
```

Ausgabe:

```
S = 210          P = 2.43290E+18
```

8.05 Programm:

```
100 PRINT " X","1/X"
110 X = -.6
120 PRINT X,
130 IF X = 0 THEN 180
140 PRINT 1/X
150 X = X+.2
160 IF X <= .6 THEN 120
170 STOP
180 PRINT "NICHT DEFINIERT"
190 GOTO 150
```

Ausgabe:

```
 X              1/X
-.6            -1.66667
-.4            -2.5
-.2            -5
 0             NICHT DEFINIERT
 .2             5
 .4             2.5
 .6             1.66667
```

8.06 Programm:

```
110 PRINT " X"," E↑X"
120 READ X
130 I = 0
140 Z = X
150 Y = 1+X
160 I = I+1
170 Z = Z*X/(I+1)
180 Y = Y+Z
190 IF ABS (Z/Y) > 1E-7 THEN 160
200 PRINT X,Y
210 GOTO 120
220 DATA -1, -.5, 0, .5, 1, 3, 5, 7
```

Ausgabe:

```
 X              E↑X
-1             .367879
-.5            .606531
 0             1
 .5            1.64872
 1             2.71828
 3             20.0855
 5             148.413
 7             1096.63

OUT OF DATA  IN 120
```

8.07 a) Schleife wird einmal durchlaufen, sie ist daher überflüssig.

b) Unendliche Schleife, weil die Schrittweite zu klein ist.

c) Bei STEP 1 wird Endwert −10 nicht erreicht.

8.08

```
10 PRINT "X(GRAD)","X(BOGEN)","SIN(X)","SIN(X+30)"
20 U = 57.2957795
30 FOR X = 5 TO 90 STEP 5
40 PRINT X,X/U, SIN(X/U), SIN((X+30)/U)
50 NEXT X
```

8.09

```
10 PRINT "ABZINSUNGSFAKTOREN"
20 PRINT
30 PRINT "N JAHRE","ABZINSUNGSFAKTOR"
40 FOR N = 1 TO 10
50 PRINT N, 1/1.05↑N
60 NEXT N
```

8.10

```
5  PRINT " X"," 1/X"
10 FOR X = -.6 TO .6 STEP .2
20 PRINT X,
30 IF X = 0 THEN 60
40 PRINT 1/X
50 GOTO 70
60 PRINT "NICHT DEFINIERT"
70 NEXT X
```

8.11 Programm:

```
100:   #####
110 Z=RND(1)
120 FOR I=1 TO 3
130 FOR J=1 TO 8
140 PRINTUSING 100, INT(10↑4*RND(0));
150 NEXT J
160 PRINT
170 NEXT I
```

Ausgabe:

```
 4735    4425    4988    3731    9213    1239     868     406
 3410    4688     297    7544    4985    2428     931    2800
 1593    2116     426    3832     889    1406    6439    8295
```

8.12

```
1:##      ##
2: N  K = 0    1    2    3    4    5    6    7    8    9
3: ####
100 PRINTUSING 2
110 FOR N = 1 TO 9
120 B = 1
130 PRINTUSING 1, N; B;
140 FOR K = 1 TO N
150 B = B*(N-K+1)/K
160 PRINTUSING 3, B;
170 NEXT K
180 PRINT
190 NEXT N
```

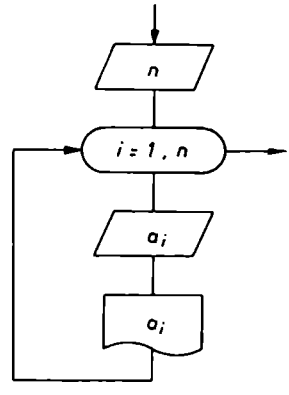
9.01

c) Index -2 ist nicht zulässig.
e) ; ist kein zulässiges Trennzeichen von Indizes.
f) Dreifach indizierte Variablen sind in BASIC nicht zulässig.

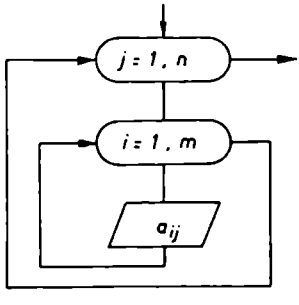
9.02

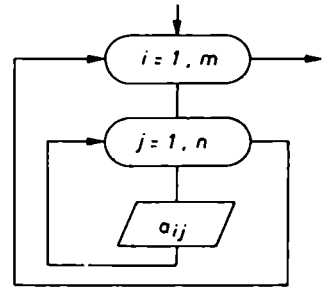
a) ; statt ,
b) 3 Indizes
c) Vorgeschriebene Form: D (natürliche Zahl)
d) Feld zu groß!

9.03



9.04





9.05

```
100:#####.###   ###   ###.##   ##.##   ###.##
110:      X(J)   F(J)      U        PHI       F *
120 DIM X(50),F(50)
130 READ X(1),F(1),B,K
140 PRINT
150 PRINTUSING 110
160 N = F(1)
170 S = F(1)*X(1)
180 Q = S*X(1)
190 FOR J = 2 TO K
200 READ F(J)
210 N = N+F(J)
220 X(J) = X(J-1)+B
230 S = S+F(J)*X(J)
240 Q = Q+F(J)*X(J)↑2
250 NEXT J
260 M = S/N
270 S = SQR((Q-S↑2/N)/(N-1))
280 C = N*B/S
290 FOR J = 1 TO K
300 U = (X(J)-M)/S
310 P = EXP(-U↑2/2)/2.50662828
320 PRINTUSING 100,X(J),F(J),U,P,C*P
330 NEXT J
340 PRINT
350 PRINT "K ="JK,"N ="JN,"M ="JM,"S ="JS
360 GOTO 130
370 DATA 827,5,1,8, 7,24,9,2,4,0,1
```

9.06

```
1:##      ##
2: N  K = 0    1    2    3    4    5    6    7    8    9
3: ####
100 PRINTUSING 2
105 B(0,0) = 1
110 FOR N = 1 TO 9
120 B(N,0) = 1
130 PRINTUSING 1, N; B(N,0);
140 FOR K = 1 TO N
150 B(N,K) = B(N-1,K-1)+B(N-1,K)
160 PRINTUSING 3, B(N,K);
170 NEXT K
180 PRINT
190 NEXT N
```

13.01
```
1ØØ  Y=SQR(1ØØ-X↑2)
3ØØ  DATA  -1Ø, 1Ø
```
Ausgabe:
FLAECHENINHALT= 157.Ø79

13.02
```
1ØØ  Y=SIN(X)
3ØØ  DATA  Ø, 3.14159265
```
Ausgabe:
FLAECHENINHALT= 1.99999

13.03
```
1ØØ  Y=(SIN(X))↑2
3ØØ  DATA  Ø, 3.14159265
```
Ausgabe:
FLAECHENINHALT= 1.57Ø8

LITERATUR

Helmut Aigner, „Mit dem Computer per Du", Verlag W. Braumüller (Wien-Stuttgart)

Herbert O. Ramp, „BASIC-Praxis", R. Oldenbourg Verlag (Wien-München)

Handbücher der Firma Honeywell-Bull

Lehrzielbank für das Berufsbildende Schulwesen
Elektronische Datenverarbeitung für Höhere technische Lehranstalten,
Experimentalfassung 1. Teil: BASIC. Bundesministerium für Unterricht und Kunst
Mitarbeiterteam: Helmut Aigner, Rudolf Bazant, Gottfried Demmelbauer,
Herbert Ramp

Wichtige Konstanten

$\pi = 3.14159265$

$2\pi = 6.28318531$

$4\pi = 12.5663706$

$\frac{4\pi}{3} = 4.18879020$

$\sqrt{2\pi} = 2.50662827$

$\sqrt{2} = 1.41421356$

$\sqrt{3} = 1.73205081$

$\lg e = \frac{1}{\ln 10} = 0.434294482$

$\mathrm{lb}\, e = \frac{1}{\ln 2} = 1.44269504$

$1\ \mathrm{rad} = 57.2957795°$

$= 3437.74677'$

$= 206264.806''$

$1° = 1.74532925\mathrm{E}{-2}\ \mathrm{rad}$

$1' = 2.90888209\mathrm{E}{-4}\ \mathrm{rad}$

$1'' = 4.84813681\mathrm{E}{-6}\ \mathrm{rad}$

REGISTER

Absolutwertfunktion 49
Abzinsungsfaktor 84
Aktualparameter 111
ALGOL 18
allgemeine Operation 65
alpha(nu)merische Information 32, 37
Altgrad 28, 49
Anschaltzeit 25
Anweisung 12, 19
Arbeitsspeicher 15
Archimedes-Näherungsverfahren 74
Arkusfunktionen 50, 115, 121
ATN 50
Auflösungsvermögen des Rechners 33
Ausgabe 17, 61, 65
Ausgabepuffer 15
Ausgleichsparabel 108

BASIC-Funktionen 48
BASIC-Term 39
bedingter Sprung 68
Bibliotheksspeicher 15
Binomialkoeffizienten 84, 90
Blattschreiber 18, 65
Bogenmaß 28, 49, 115
BYE 17, 25, 123, 139

CALL 117
Cardano-Formeln 120
CAT 139
Compiler 17
CON 102
COS 49

DATA-Zeile 21, 26, 60
Daten 11
Datenerschöpfung 67
Datenfelder 85
DEFFN-Anweisung 57
DIM-Anweisung 86, 96
DIMENSION ERROR 97
DIVISION BY ZERO 25
Doppelpunktanweisung 63
Dreiecksberechnung 16, 51, 59
Dreiwegeverzweigung 70

e 49, 76
EDIT-Befehle 140
Eingabe 60, 65
Eingabepuffer 15
Einheitsmatrix 100
END 38
endlose Schleife 81
Ergibt-Anweisung 42
Euklid-Algorithmus 112
EXP 49
explizite Dimensionierung 87

Feld 86
Fixkommazahlen 33
Flächeninhalt 45, 126
Flußdiagramm 65
Formalparameter 58, 110
FOR-Anweisung 77
formatisierte Ausgabe 63
FORTRAN 18
Funktionen 48

Ganzzahlen 32
Ganzzahl — Unendlich 32
Gauß-Jordan-Algorithmus 135
Gauß-Normalgleichungen 108
gestrecktes Programm 66
Gleichheit zweier Matrizen 98
Gleichungssystem 106, 135
Gleitkommazahlen 33
GOSUB 111
GOTO 25, 37
Grenzstelle 65
größter gemeinsamer Teiler 112

Hauptprogramm 111
Hauptspeicher 12
Horner-Schema 87

IDN 100
IF-Anweisung 68
indizierte Variable 85
induktive Schleife 71
Initialisierung 35
Inkrement 77
INPUT 61
INT 53
Intervallschachtelung 124
inverse Matrix 104
Iteration 74
iterative Schleifen 72, 76

Kehrmatrix 104
KEY 139
Klammern in BASIC 40
Konnektor 66
Konstanten 32, 37
konstante Matrix 102

Kreisfunktionen 49
kubische Gleichung 120

Laufvariable 77
Leeranweisung 19
LET 42
lineare Gleichung 75, 147
lineares Gleichungssystem 106, 135
LIS 139
LISTNH 20, 139
Lochstreifen 18
LOG 49

Mark III 56, 121
Matrizenoperationen 95
Mehrfachzuweisung 44
Mittelwert 79

näherungsweise Auflösung von Gleichungen 124
natürlicher Logarithmus 49
NEARLY SINGULAR MATRIX 104
Neugrad 28
NEW 18, 139
NEXT 77
Normalverteilung 90
Nullmatrix 102
numerische Information 32
numerische Integration 126
numerische Variablen 34

Objektprogramm 17, 26
ODER-Verknüpfung 70
off-line-Peripherie 12
OLD 16, 139
on-line-Peripherie 12
OUT OF DATA IN ... 26, 61
Output 24

Platzreservierung 86
Polynom 87
Potenzierung 21, 35, 36
PRINT 19, 61
PRINTUSING 63, 84
PRINT TAB 91
Produkt von *n* Zahlen 78
Produktmenge 82
Programm 12
Programmablaufplan 65
Programmzeile 19

quadratische Gleichung 118
Quadratwurzelfunktion SQR 21, 27, 49
Quellenprogramm 17, 26
Querschnittswerte 45, 132

READ 21, 26, 60
READY 16
Rechenoperationen 35
Rechenwerk 12
Rechenzeit 17, 25
Rechnergenauigkeit 33
Rechner — Unendlich 33
reelle Nullstellen 124
Relationszeichen 69
relative Häufigkeit 89
REN 139
REM 38
REP 140
RETURN 111
RETURN-Taste 16, 19
RND 54
RUN 17, 19, 20, 23, 26, 139
Runden von Dezimalzahlen 53

SAVE 140
Schablonen 63
Schalter 130
Scheinvariable 58
Schleife 66, 71
Schleifennest 81
Schlußdaten 89
Schriftzeichen in BASIC 32
Schrittweite 77
SCR 140
SGN 49
Sehnen-Trapez-Regel 126
signifikante Ziffer 33
Signumfunktion 49
SIN 49, 92
Skalarprodukt 80
Sonderzeichen 32
Sortierverfahren 128
Spaltenvektor 95
Sprunganweisungen 25, 37, 68
SQR 21, 27, 49
SQUARE ROOT OF NEGATIVE NUMBER 27
Standardabweichung 79
Standardfunktionen 48
statistische Maßzahlen 79
STEP 77
Steuerwerk 12
STOP 38, 115
sukzessive Schleife 72
Summen 78
SYS 140
Systembefehle 139
Systembibliothek 15

TAB 91
TAN 49
TAP 18, 25, 140
Term 39
Terminal 15
Time-Sharing 14
Trägheitsmoment 45, 132
transponierte Matrix 99

Übergangsstelle 66
unbedingte Sprunganweisung 25, 37
UND-Verknüpfung 70
UNS 140
Unterprogramm 66, 110

Variable 22, 34, 37
Variablenfelder 85
Varianz 79
Vektor 87
vereinbarte Funktionen 57
Verzweigungen 66, 68
Vorzeichenfunktion 49

Weiche 130
Wertzuweisung 42, 44
Widerstandsmoment 45
Winkelumrechnungen 28, 115

Zahlenbereich des Rechners 33
Zahlendarstellungen 32
zählergesteuerte Schleife 71, 77
Zeichnen von Kurven 91
Zeilennummer 19, 23
Zeilenvektor 95
Zeilenvorschub 19
Zentraleinheit 11
ZER 102
Zufallszahlen 54
Zuweisung 42
zweifach indizierte Variablen 85
Zweiwegeverzweigung 70

HERBERT O. RAMP

BASIC-PRAXIS

Zweite, verbesserte und erweiterte Auflage 1975

Dieses Buch wendet sich, als Folgeband zum vorliegenden Buch, an Leser, die ihre Programmierkenntnisse erweitern, vertiefen und verfeinern wollen. Das Leitmotiv des Buches ist: „Wie programmiere ich besser?"
Ein neuer Mark III-BASIC gewidmeter Teil bespricht Unterschiede zu Mark I-BASIC, führt neue, an alphamerischen Größen operierende Befehle ein und zeigt verschiedene Syntax-Feinheiten auf.
Übungsfragen mit Lösungen, zahlreiche ausgearbeitete Musterprogramme, meist für direkte praktische Anwendung, machen das Buch für ein Selbststudium geeignet.

In Vorbereitung für 1978:

JULIUS SCHÄRF / TOBIAS SCHLÄPFER

BASIC-ÜBUNGEN FÜR ANFÄNGER

Dieses Buch wird — im Anschluß an „BASIC für Anfänger" — dem Lernenden eine gute Hilfe sein und ihm Anregungen zum Erstellen von eigenen BASIC-Programmen bieten.

ELEKTRONISCHE TASCHENRECHNER

Reihe: Fortbildung durch Selbststudium

HERAUSGEGEBEN VON JULIUS SCHÄRF

JULIUS SCHÄRF / ROBERT STRECHA

UPN- Taschenrechner in Schule und Praxis

190 Seiten, flex. Kunststoff S 144.—

JULIUS SCHÄRF / HELMUT SCHIERER / ROBERT STRECHA

Die Taschenrechner SR-50A und SR-51A in Schule und Praxis

142 Seiten, flex. Kunststoff S 124.—

JULIUS SCHÄRF / HELMUT SCHIERER / ROBERT STRECHA

Der Taschenrechner SR-51-II in Schule und Praxis

134 Seiten, flex. Kunststoff S 108.—

JULIUS SCHÄRF / HELMUT SCHIERER / ROBERT STRECHA

Die Taschenrechner TI-30, TI-45 und SR-40 in Schule und Praxis

102 Seiten, flex. Kunststoff S 82.—

JULIUS SCHÄRF / ROBERT STRECHA / WERNER BARON

Programmieren mit dem Taschenrechner SR-56

Ca. 164 Seiten, flex. Kunststoff ca. S 140.—

JULIUS SCHÄRF / HELMUT SCHIERER / WERNER BARON

Programmieren mit dem Taschenrechner TI-57

Ca. 144 Seiten, flex. Kunststoff ca. S 132.—

JULIUS SCHÄRF / HELMUT SCHIERER / WERNER BARON

Programmieren mit dem Taschenrechner TI-58 und TI-59

In Vorbereitung

JULIUS SCHÄRF / HELMUT SCHIERER / WERNER BARON

Programmieren mit dem Taschenrechner HP-25

In Vorbereitung

R. OLDENBOURG VERLAG WIEN MÜNCHEN

Diese Buchreihe verhilft zu einer schnellen und sicheren Aneignung von Wissen, das für den beruflichen Aufstieg in der heutigen Leistungsgesellschaft erforderlich ist.
Anhand von vollständig durchgeführten Musterbeispielen dringt der Leser schrittweise in das Wissensgebiet ein. Viele Übungen mit angegebenen Lösungen geben ihm die Möglichkeit, die erworbenen Kenntnisse zu überprüfen und schon bald die ihm aufgetragenen Arbeiten zu bewältigen.

Dieses Buch ist für Anfänger auf dem Gebiet der Datenverarbeitung geschrieben. Besondere mathematische Kenntnisse werden nicht vorausgesetzt.

Der Verfasser hat aus diesem Grund die weitverbreitete Programmiersprache BASIC gewählt, die in einigen Stunden leicht erlernbar ist.

Für die Neuauflage dieses Buches wurden Vereinfachungen der Sprache BASIC berücksichtigt. Zusätzlich werden am Beginn jedes Abschnitts die Lernziele angegeben. Um die Leser — auch Lernwillige, denen keine Datenverarbeitungsanlage zur Verfügung steht — schon beim Studium an das für sie neue Schriftbild des Terminals zu gewöhnen, werden Programme und Ausgabeprotokolle fotografisch wiedergegeben.

FORTBILDUNG DURCH SELBSTSTUDIUM

R. Oldenbourg Verlag Wien München

ISBN 3-7029-0115-9 R. Oldenbourg Wien
ISBN 3-486-39203-6 R. Oldenbourg München